ANCORA PIÙ RICCA!!! 132 PAGINE

IMPARA A PROGRAMMARE IL CELLULARE: E' UN GIOCO!

Periodicità mensile • APRILE 2003 • ANNO VII, N.4 (68) RIVISTA+LIBRO+CD €7,70 anziché €7,90

Poste Italiane • Spedizione in a.p. - 45% • art. 2 comma 20/b legge 662/96 - AUT. N. DCDC/033/01/CS/CAL

IO PROGRAMMO

IN REGALO

Sicurezza e HACKER: spiare i PC in rete

- ✔ Controllo oltre ogni limite con i raw socket
- ✔ Realizziamo passo passo uno sniffer
- ✔ Tecniche di crittografia e firma digitale

SDK completo sul CD

Microsoft DirectX 9.0

Mettiamo in pratica i nuovi effetti luminosi

Video Sorveglianza sul Web

Ora è facile, con i componenti Macromedia Server

Extreme Programming

Le tecniche per mantenere il codice sempre aggiornato

SISTEMA

- Come realizzare l'help delle nostre applicazioni
- Utilizzare i componenti Delphi in C++
- Excel e VBA: imparare per esempi

WEB PROGRAMMING

- Web Services e Database: interrogazioni e sicurezza
- La costruzione di applicazioni Web con i Design Pattern

ELETTRONICA & INTERNET

- Un progetto completo per controllare apparecchi elettronici dal Web

CORSI

- *MATLAB*: risoluzione di sistemi lineari
- *VISUAL BASIC*: integrare l'Help in un progetto
- *VB.NET*: guida all'uso degli oggetti
- C#: l'overload degli operatori
- C++: i template

FAQ & TIPS

60 SOLUZIONI pronte all'uso

ISSN 1128-594X

30068

9 771128 594009

MULTIMEDIA: la realizzazione degli oggetti in Lightwave

Contents

Anno VII - n. 4 (68) Aprile 2003

Fiocco rosa per Edizioni Master

Come avrete notato, questo è il secondo mese che con ioProgrammo non trovate più l'inserto Programmare il WEB. Molti lettori ci hanno scritto lamentandosi di questa assenza... ma non si è trattato di un passo indietro, bensì di un grande passo avanti: la nascita di una nuova rivista! Questo mese, in edicola trovate il primo numero di *<tag />*, un'intera rivista tutta dedicata allo sviluppo orientato al web, nata proprio grazie al grande successo che Programmare il web ha riscosso. Di Programmare il WEB troverete lo stesso entusiasmo ed alcune delle firme più prestigiose e, oltre alla programmazione più propriamente detta, sarà dedicato ampio spazio a tutto quanto concerne la produzione di contenuti e interfacce per il Web. Il testimone passa dunque all'ottimo Thomas Zaffino che ha coordinato il lavoro dei migliori professionisti del Web: ognuno ha messo a disposizione la propria esperienza e ha costruito delle vere e proprie mini-guide che, mettendo da parte i sofismi e gli "Hello World", conducono gli sviluppatori ad ampliare le proprie conoscenze con esempi concreti e immediatamente riutilizzabili. Nello stile che ha decretato il successo di ioProgrammo.

P.S. Queste note le leggerete all'inizio della primavera, ma ora che scrivo è ancora inverno e soffiano freddissimi i venti della guerra. Anche noi, anche qui, vogliamo dire Pace.

raffaele@edmaster.it

Raffaele del Monaco

ioProgrammo

Anno VII - N.ro 4 (68) - Aprile 2003 - Periodicità: Mensile
Reg. Trib. di CS al n.ro 593 - Cod. ISSN 1128-594X
E-mail: ioprogrammo@edmaster.it
http://www.edmaster.it/ioprogrammo

Dir. Editoriale Massimo Sesti
Dir. Responsabile Romina Sesti
Product Manager Antonio Meduri
Editor Gianfranco Forlino
Coordinatore redazionale Raffaele del Monaco
Redazione Rossella Berardi, Antonio Pasqua, Thomas Zaffino
Collaboratori M. Autiero, L. Buono, M. Canducci, S. Fago, E. Florio, S. Cristiano, M. Del Gobbo, R. Lombardo, L. Nanni, S. Marano, A. Marroccelli, C. Pelliccia, P. Perrotta, G. Pinelli, F. Sara, S. Serra, L. Spuntoni, G. Uboldi, F. Vaccaro, I. Venuti.
Segreteria di Redazione Veronica Longo

REALIZZAZIONE GRAFICA CROMATIKA Srl
C.da Lecco, zona ind. - 87030 Rende (CS)
Tel. 0984 8319 - Fax 0984 8319225
Coord. grafico: Paolo Cristiano
Coord. tecnico: Giancarlo Sicilia
Impaginazione elettronica: Aurelio Monaco

"Rispettare l'uomo e l'ambiente in cui esso vive e lavora è una parte di tutto ciò che facciamo e di ogni decisione che prendiamo per assicurare che le nostre operazioni siano basate sul continuo miglioramento delle performance ambientali e sulla prevenzione dell'inquinamento"

Certificato UNI EN ISO 14001 N. 9191 CRMT

PUBBLICITÀ Edizioni Master S.r.l.
Responsabile Vendite Ernesto Redaelli
Agenti Vendita Elisabetta Februo, Serenella Scarpa, Cornelio Morari
Segreteria Ufficio Vendite Daisy Zonato

Via Cesare Correnti, 1 - 20123 Milano
Tel. 02 8321612 - Fax 02 8321764
e-mail: advertising@edmaster.it
EDITORE Edizioni Master S.r.l.
Sede di Milano: Via Cesare Correnti, 1 - 20123 Milano
Tel. 02 8321482 - Fax 02 8321699
Sede di Cosenza: C.da Lecco, zona ind. - 87030 Rende (CS)
Amministratore Unico: Massimo Sesti
Responsabile Amministraz. e Finanza: Benedetto Celsa
Produzione e Logistica: Michele Carere
Diffusione: Alessandra Cervello
Marketing: Giuseppina Bruno, Leonardo Petrone, Antonio Meduri

ABBONAMENTO E ARRETRATI
Italia: Costo abbonamento annuale basic (11 numeri) € 84,70, Promozione Sconto 50% € 42,50. Costo abbonamento Plus (11 numeri + 6 libri) € 128,50, promozione sconto 50% € 64,50. **Estero:** Costo abbonamento annuale (11 numeri) € 169,40, costo abbonamento Plus (11 numeri + 6 libri) € 257,00.
Costo arretrati (a copia): il doppio del prezzo di copertina + € 5,16 spese (spedizione con corriere). Prima di inviare i pagamenti, verificare la disponibilità delle copie arretrate allo 028321482. La richiesta contenente i Vs. dati anagrafici e il nome della rivista, dovrà essere inviata via fax allo 028321699, oppure via posta a EDIZIONI MASTER via Cesare Correnti, 1 - 20123 Milano, dopo avere effettuato il pagamento, secondo le modalità di seguito elencate:

- cc/p n.16821878 o vaglia postale (inviando copia della ricevuta del versamento insieme alla richiesta);
- assegno bancario non trasferibile (da inviarsi in busta chiusa insieme alla richiesta);
- carta di credito, circuito VISA, CARTASI', MASTERCARD/ EUROCARD, (inviando la Vs. autorizzazione, il numero della carta, la data di scadenza e la Vs. sottoscrizione insieme alla richiesta).

SI PREGA DI UTILIZZARE IL MODULO RICHIESTA ABBONAMENTO POSTO NELLE PAGINE INTERNE DELLA RIVISTA. L'abbonamento verrà attivato sul primo numero utile, successivo alla data della richiesta.
Sostituzioni: Inviare il CD-Rom difettoso in busta chiusa a:

Edizioni Master Servizio Clienti - Via Cesari Correnti, 1 20123 Milano

Assistenza tecnica: ioprogrammo@edmaster.it
Servizio Abbonati:
tel.02 8321482
@ e-mail: servizioabbonati@edmaster.it

Stampa: Elcograf Industria Grafica (LC)
Stampa CD-Rom: Disctronics Italia (MI)
Distribuzione per l'Italia: Parrini & C S.p.A. - Roma

Finito di stampare nel mese di Marzo 2003

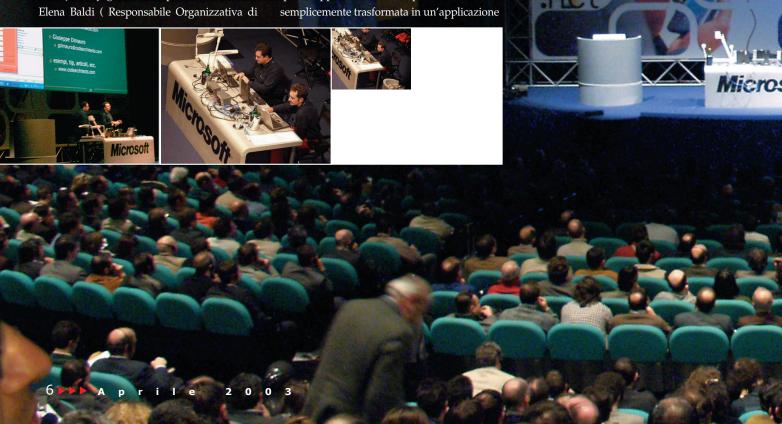
Edizioni Master edita:
Idea Web, Go!OnLine Internet Magazine, Win Magazine, Quale Computer, DVD Magazine, Office Magazine, ioProgrammo, Linux Magazine, Softline Software World, MPC, Discovery DVD, Computer Games Gold, inDVD, I Fantastici CD-Rom, PC VideoGuide, Il CD-Rom di Idea Web, I Corsi di Win Magazine, Le Collection.

Dev Days 2003

Il 31 gennaio scorso, Microsoft ha organizzato a Roma una edizione speciale del Dev Days cui ha partecipato Bill Gates fondatore e Chief Software Architect della casa di Redmond. ioProgrammo era lì per voi!

L'intervento di Bill Gates è previsto per il primo pomeriggio ma, già dalla mattina, c'è grande attesa fra tutti i presenti per l'arrivo dell'uomo più ricco del mondo, per colui che più di ogni altro rappresenta la rivoluzione informatica nel mondo. La mattina è riservata all'incontro con tutte le più alte cariche dello stato: Ciampi, Berlusconi, il presidente del Senato Pera. Le misure di sicurezza si annunciano imponenti ma tutto si svolge nella più assoluta calma, come se l'idea di un attentato non potesse nemmeno sfiorare la placida calma di una Roma livida di freddo e bella e indifferente come sempre. All'ingresso della conferenza, i partner dell'evento distribuivano gadget e informazioni con grande solerzia e, con una punta di orgoglio, dobbiamo dire che il regalo più gettonato erano le copie di ioProgrammo distribuite allo stand della ObjectWay grazie allo splendido lavoro di Elena Baldi (Responsabile Organizzativa di ObjectWay University). Alla fine, il colpo d'occhio nella sala della conferenza sarà davvero impressionante: centinaia di sviluppatori che sfogliano la nostra rivista preferita!

HANDS ON MORNING

La mattina è dedicata ad una interessante sessione prettamente tecnica in cui gli ottimi Francesco Balena e Giuseppe Dimauro hanno illustrato con grande entusiasmo le mille possibilità offerte dal Framework .NET e da Visual Studio. Balena, da vero mattatore, ha costruito con poche righe di codice VB.NET un'applicazione stand alone che interrogava una base di dati, ponendo l'accento sull'importanza di scrivere del codice indipendente dal tipo di DB da interrogare. Grazie a piccoli accorgimenti e all'estrema flessibilità di Visual Studio .NET, questa applicazione è stata poi velocemente e semplicemente trasformata in un'applicazione Three Tier nella sessione successiva, tenuta da Dimauro. Dopo aver simpaticamente duellato a colpi di fioretto sulla presunta superiorità di C# rispetto a VB.NET, Dimauro ha esteso l'applicazione realizzata pochi istanti prima da Francesco Balena, realizzando una completa

soluzione Three Tier, arricchendola con la gestione degli aspetti riguardanti la sicurezza. Balena e Dimauro hanno quindi introdotto i generatori di codice Template-based e la programmazione by-contract: una serie di regole di programmazione che superano la semplice prototipizzazione delle funzioni. Attraverso la definizione di intere classi o di un sottoinsieme di API, è possibile risolvere elegantemente e con notevole "robustezza" una ampia classe di problemi. Le linee guida di questo approccio alla programmazione si possono riassumere in due punti: ogni interfaccia deve risolvere una particolare problematica e, se ci si trova nella condizione di non saper dare un nome ad una interfaccia, vuol dire che le abbiamo assegnato troppi compiti ed è dunque il caso di ripensare l'impostazione dell'interfaccia stessa, magari dividendola in più parti. La programmazione by-contract ha il vantaggio di consentire una implementazione reale sganciata e indipendente dall'effettivo utilizzatore, dando così la possibilità di un più intensivo e vantaggioso utilizzo del polimorfismo. La sessione mattutina si è conclusa con un interessante esempio di compilazione on-the-fly, attraverso l'utilizzo di oggetti VBCodeCompiler e CsharpCodeCompiler: .NET è stato per così dire "piegato" ad una sorta di scripting che consentiva di modificare al volo il comportamento dell'applicazione. Un sapiente utilizzo di queste tecniche consente di introdurre nelle applicazioni funzionalità che vanno dall'interpretazione di espressioni alla esecuzione di macro, per arrivare alla personalizzazione dell'applicazione in relazione all'utente. Insomma, gli sviluppatori che hanno avuto la fortuna di assistere a questa presentazione hanno avuto modo di toccare con mano le grandi potenzialità di .NET.

POMERIGGIO D'ATTESA

Dopo la breve pausa per la colazione, il pomeriggio è cominciato all'insegna di una tensione palpabile per l'imminente arrivo del capo carismatico di Microsoft. La sala è andata riempiendosi fino all'inverosimile ed è iniziato uno splendido intervento di Francesco Albano che, una volta di più, si è confermato come uno dei migliori speaker in circolazione. Con la sua solita, travolgente passione, Francesco ha ripercorso le tappe che hanno portato alla definizione dello standard XML. Con un delizioso paragone fra HTML ed il mito di Narciso, ha sottolineato quanto il vecchio linguaggio di ipertesto fosse orientato esclusivamente alla forma e assolutamente "indifferente" al contenuto. Con XML è stato possibile dare vita alle informazioni e renderle disponibili e utili in un modo assolutamente impensabile con HTML. Per esemplificare i concetti di condivisione e facilità di accesso resi possibili da XML e dai Web Services, Albano si è avvalso della collaborazione di Dimauro che è tornato sul palco per dare il via ad una dimostrazione che, nella sua semplicità, è stata di notevole impatto per il pubblico presente in sala. Poche righe di codice .NET hanno permesso di ottenere una completa applicazione di informazioni turistiche: scelta la località di partenza e di arrivo, ed i giorni in cui volevamo andare in vacanza, l'applicazione si faceva carico di interrogare (tramite Web Services, of course!) le banche dati di svariate compagnie aree, proponendoci poi una serie di soluzioni compatibili con le nostre richieste. Anche la scelta dell'albergo e la localizzazione dello stesso su una cartina (servizio erogato da MapPoint) erano implementati in questa piccola applicazione che, in sedicesimo, illustrava la rivoluzione che stiamo vivendo.

DIGITAL DECADE

Sul finire dell'intervento di Francesco Albano, la sala ha cominciato ad agitarsi, mentre l'apparizione di una nutrita serie di agenti della sicurezza preannunciava l'imminente arrivo della star della giornata. Proprio come una star

Hollywoodiana, Bill Gates è stato accolto da una impressionante batteria di flash: il clamore ha per un po' coperto la tensione del momento, poi un silenzio curioso e attento ha accolto le prime parole del presidente di Microsoft. I prossimi dieci anni saranno ricordati come il Digitale Decade, il decennio digitale: questa è la convinzione di Gates e, da questo assunto, parte la descrizione della "visione" tecnologica di Microsoft. I prossimi anni saranno "fondanti" per tutto quello che sarà l'avvenire dell'informatica nel mondo, dai Tablet PC (l'attuale cavallo di battaglia di Microsoft) ai sistemi digitali da indossare, l'idea di Gates è che molte delle tecnologie che fino a pochi anni fa si potevano solo sognare, siano in realtà già presenti (parafrasando i fratelli Cohen, si potrebbe dire: "the future is now"). Il compito che investe i prossimi dieci anni è quello di armonizzare le tecnologie già disponibili: digital imaging, Internet, home-cinema e più in generale tutto quanto riguardi la domotica, dovrà convergere all'interno di un'unica infrastruttura aperta che consenta il dialogo fra tutti i soggetti e le apparecchiature elettroniche che entreranno (e in parte sono già entrati) nella nostra vita. Gates ha dunque illustrato la centralità di XML e della ricerca di protocolli condivisi in questo scenario: rompendo una antica tradizione di isolamento, Microsoft ha contribuito in maniera determinante alla nascita del Web Services Interoperability Organization in cui, insieme ad altri giganti dell'informatica del calibro di IBM, si stanno mettendo le basi per un nuovo modo di programmare e condividere le risorse. E' indubbio che Microsoft stia investendo enormemente nei Web Services e in XML che rappresentano entrambi parte integrante di tutte le piattaforme Microsoft: "XML rappresenta le fondamenta di tutta l'informatica", queste le parole con cui Gates ha definito un passaggio da cui non si tornerà più indietro.

IL COLPO DI TEATRO

Proprio mentre il Presidente di Microsoft descriveva le delizie attuali e future dei Web Services, con un vero e proprio colpo di teatro Francesco Albano è tornato sul palco e un piccolo cilindro ha cominciato a ruotare svelando una lavatrice… Tra il generale stupore, Francesco ha imbracciato la nuova arma d'assalto di Microsoft, un tablet PC (rigorosamente e pesantemente wired…) che ha cominciato ad usare a mo' di telecomando. Sullo schermo gigante è comparsa la piantina di una casa con alcuni elettrodomestici evidenziati, ovviamente(!) uno di questi era il simbolo di una lavatrice. Sfiorando lo schermo con la stilo, Francesco ha impostato un programma di lavaggio e ha avviato la lavatrice. Come è stato possibile? Tramite Web Service, of course! E grazie allo splendido lavoro compiuto da un gruppo tutto italiano (La Thinkware di Napoli) che è riuscito a cablare tutta la logica necessaria ad un Web Service in un singolo chip. Sotto lo sguardo attento e divertito di Bill Gates, questo piccolo show è continuato con la visione di un film sul grande schermo, a rappresentare la possibilità includere l'Home Theater nella casa del (prossimo) futuro immaginata da Microsoft. Manco a dirlo, la visione del film è stata interrotta da un messaggio della lavatrice che segnalava un problema (sempre tramite Web Services) e richiedeva l'intervento dell'operatore… pardon, del padrone di casa!

SICUREZZA

In sintesi, anche la spettacolare performance dedicata alla lavatrice e alla domotica era incentrata sui Web Services e, al momento di riprendere la parola, Bill Gates è tornato a parlare proprio dei Web Services, sottolineando questa volta le attuali carenze dello standard: sicurezza e transazioni. Ha dunque annunciato che saranno proprio queste le linee guida dei prossimi sviluppi che coinvolgeranno i Web Services. Gates ha sottolineato che proprio la sempre maggiore presenza della tecnologia nella vita e nel lavoro di ognuno, richiede un'attenzione sempre maggiore ai problemi riguardanti la sicurezza ed ha dunque introdotto l'argomento Palladium. Gates ha spiegato che allo stato attuale non c'è modo di garantire che il software ed il sistema operativo siano esattamente come il produttore li ha concepiti: segretezza dei dati personali e solidità delle applicazioni sono dunque esposte a grandi rischi. Solo una stretta collaborazione fra hardware e software può garantire gli alti livelli di sicurezza ormai necessari e Gates ha dunque spiegato le finalità del Trusted Computing Platform Alliance che sono proprio quelle di creare un complesso hardware/software che garantisca la autenticità del software in esecuzione. Palladium è dunque il lato software di questa piattaforma che Microsoft ha intenzione di incorporare in tutte le future versioni di Windows. Il sistema di sicurezza di Palladium prevede una sorta di catena che dal chip, al sistema operativo, alle applicazione fino ad arrivare ai dati sensibili dell'utente, garantisce l'integrità e la sicurezza delle informazioni. Gates è quindi passato ad illustrare "the next big thing" di Microsoft: la tecnologia Spot. Grazie ad una spinta miniaturizzazione e ad un attento lavoro di ingegnerizzazione volto alla riduzione dei consumi, questa tecnologia consente la realizzazione di dispositivi della dimensione di un orologio da polso in grado di ricevere informazioni (via radio, FM) e filtrarle secondo i gusti e le esigenze degli utenti. Gates ha concluso l'intervento accennando all'intelligenza artificiale e al riconoscimento vocale, due tecnologie che non hanno raggiunto i risultati che si attendevano, ma su cui Microsoft investe ancora molto.

Insomma, ci aspetta una decade di sviluppo impetuoso, ma non equivocate: il futuro è già arrivato!

Raffaele del Monaco

Ringraziamenti

Per le foto di questo articolo desidero ringraziare Carlo Pinasco (Senior .NET Architect di Microsoft Italia) e l'eccezionale Chiara Mizzi di Imageware.

News

Windows Server 2003

Ad Aprile sarà distribuito il nuovo sistema operativo di casa Microsoft: sicuro, affidabile, scalabile

Il nuovo sistema operativo di casa Microsoft, fino a poco tempo fa conosciuto come Windows.Net Server 2003, verrà consegnato a partire dal prossimo 24 aprile col nome "Windows Server 2003". Il nuovo prodotto verrà presentato a San Francisco assieme a Visual Studio.Net 2003, appositamente progettato per realizzare applicazioni sul nuovo Windows Server 2003. Al fine di semplificarne la gestione e la diffusione, la Microsoft ha progettato il nuovo prodotto basandosi sui punti di forza di Windows 2000. Dal confronto tra i due prodotti è emerso che per molte caratteristiche il nuovo prodotto offre prestazioni migliori (fino al 140%) del precedente. Il principale obiettivo del nuovo sistema operativo è la sicurezza, inoltre presenta notevoli vantaggi in termini di produttività, affidabilità, connettività e scalabilità. Il nuovo prodotto supporta cluster di server fino ad otto nodi e garantisce un'ottima gestione del "failover"(se c'è un guasto un nodo, subentra immediatamente un altro nodo che si preoccupa di fornire il servizio richiesto). L'ambiente è in grado di salvaguardare, in modo sicuro, le informazioni relative ad aziende e clienti migliorando la produttività delle aziende e dei loro dipendenti. Tra le nuove caratteristiche introdotte in Windows Server 2003 vanno citate: il Common Language Runtime e Internet Information Services 6.0. Il primo verifica la correttezza delle applicazioni, le autorizzazioni di protezioni e riduce notevolmente eventuali bug causati da errori di programmazione. IIS 6.0 (Internet Information Services) serve a proteggere i processi che utilizzano web services permettendo di lavorare sulle proprie applicazioni senza il timore di attacchi via Internet. Nei mesi successivi all'uscita del prodotto, la Microsoft rilascerà nuove tecnologie integrabili con Windows Server 2003 al fine di ottimizzare e migliorare ulteriormente l'utilizzo di questo nuovo sistema operativo.

www.microsoft.com

Dal BIOS verso l'EFI

Nel corso dell'Intel Developer Forum tenutosi a San Jose l'Intel ha annunciato il successore dell'ormai vecchio BIOS

Ebbene si! Dopo più di vent'anni il nostro caro BIOS potrebbe lasciarci per cedere il suo posto ad un nuovo progetto di casa Intel: EFI (Extensible Firmware Interface). L' EFI promette di ovviare a tutti i problemi e limiti del BIOS legati al fatto che varia da un computer all'altro anche se il suo compito rimane sempre lo stesso. Gli interventi di manutenzione saranno più semplici da attuare grazie all'utilizzo di un'interfaccia grafica e di risorse per diagnostiche a distanza. Anche la sua programmazione sarà più flessibile perché non sarà più scritto in Assembler ma in C , dunque sarà possibile integrare in EFI programmi autoconfiguranti e migliori strumenti di diagnostica. Naturalmente la transizione dal BIOS all'EFI non si attuerà dall'oggi al domani, per questo, almeno per i primi tempi, l'EFI potrebbe comparire affiancato dal BIOS.

http://www.intel.com/technology/efi/efi.htm

Nuovi Video MMS

Openwave e PacketVideo annunciano la possibilità di inviare video clip via MMS

Nel corso del 3GSM World Congress, tenutosi a Cannes nel mese di Febbraio, Openwave e PacketVideo hanno annunciato la possibilità di inviare video tramite MMS. Oltre a loghi e suonerie, gli utenti avranno la possibilità di scambiarsi dei veri e propri video semplicemente sfruttando l'infrastruttura standard MMS.

La tecnologia mobilemedia, utilizzata per questo nuovo servizio, consente di registrare i propri video clip ed inviarli come allegati di messaggi MMS. Il ricevente può visualizzare il filmato via streaming o download. Questa tecnologia non è altro che una codifica audio/video sviluppata da PacketVideo, integrata con la piattaforma MMSC di Openwave. Grazie all'utilizzo delle reti GPRS ed UMTS ed alla diffusione di telefonini dotati di tecnologie sempre più

evolute, si prevede che nel prossimo futuro la diffusione di MMS supererà di gran lunga quelle degli SMS.

www.openwave.com

Flash entra nei telefonini con DoCoMo

Il gigante giapponese sarà il primo a offrire servizi basati sulla tecnologia Macromedia

Flash è una delle più diffuse applicazioni per PC, essendo installata su oltre il 98% dei desktop in tutto il mondo, rappresentando lo strumento principe per la diffusione di pubblicità interattive e piccoli cartoni animati. Macromedia punta ancora più in alto e, con le nuove versioni di Flash, vuole diventare una completa piattaforma per Web Services, i cui benefici possono arrivare anche a dispositivi portatili come cellulari e palmari. Il recente accordo prevede che DoCoMo includerà un player Flash nei suoi nuovi cellulari di fascia alta (a cominciare dalla serie 505i) a partire dalla fine del 2003. Saranno contestualmente resi disponibili tutta una serie di servizi interattivi che, grazie a Flash, potranno essere interrogati senza dover passare attraverso delle noiose e confuse interfacce testuali.

L'interazione avverrà tutta per via visuale. Ad esempio, uno dei primi servizi disponibili saranno le previsioni del tempo: la scelta della località si effettuerà semplicemente cliccando la zona che desideriamo su un mappa interattiva. Per FlahMX sono già disponibili numerosi tool che consentono di adattare i contenuti esistenti a formati diversi da quelli standard di un PC, compresi i display dei telefonini. Questo consentirà di rendere subito disponibile un ampio ventaglio di applicazioni e servizi. Flash ha ottime chance anche grazie al massiccio uso di grafica vettoriale che lo caratterizza, cosa che si traduci in un'occupazione di banda minore, con vantaggi per la velocità e per le tasche degli utenti.

www.nttdocomo.com

PlayStation2 gioca on line con il Grid Computing

IBM e Sony hanno annunciato una nuova piattaforma per i giochi on line

IBM e Sony Computer Entertainment hanno annunciato un accordo per fornire agli sviluppatori un ambiente basato sul grid computing, che consentirà di realizzare e testare giochi multiplayer on line per la PlayStation2. Tutta la fase di ingegnerizzazione e test dei giochi potrà dunque avvenire direttamente attraverso una reale infrastruttura di grid computing, con un notevole risparmi di tempo e di costi rispetto alle comuni simulazioni in scala. La soluzione congiunta IBM-Sony, la Butterfly Grid, è indirizzata a risolvere i tre principali problemi che affliggono lo sviluppo di giochi multiplayer che prevedono una massiccia presenza di giocatori:

- Come garantire velocità di elaborazione e bassa latenza per tutti i giocatori.
- Come garantire che l'istante di gioco sia coerentemente "distribuito" a tutti senza errori.
- Riuscire a tenere costante la capacità computazionale, rispondendo prontamente all'ingresso e all'uscita di nuovi giocatori e alle "cadute" dei nodi.

La tecnologia Butterlfy prevede l'utilizzo di più server coordinati che reagiscono come un unico supercomputer virtuale, con la capacità spostare autonomamente i processi da una macchina all'altra. I giochi potranno dunque avere accesso a maggiori risorse e maggiore spazio su server, mano a mano che attireranno nuovi giocatori: il sistema si adatterà automaticamente alle nuove necessità computazionali.

www.butterfly.net

Google acquista Pyra Labs, dopo i newsgrup arrivano i Weblog?

Il gruppo di sviluppatori che più di tutti hanno lanciato il fenomeno Blog entrerà nel team di Google

Negli ultimi tempi i Weblog (o blog) si sono affermati come una delle principali forme di espressione su Internet: la semplicità dei newsgroup unita alla visibilità offerta dal Web hanno decretato il successo di questa tecnica di pubblishing. Google, seguendo una strategia che l'ha portato a comprare prima l'archivio di Deja.com per poi sperimentare un servizio di news, arriva ora a includere nel suo team il gruppo si sviluppatori che sta dietro al successo dei Weblog. Il sito ufficiale dei Pyra Labs (Blogger.com) conta più di un milione di utenti registrati, un milione di utenti che utilizzano il loro sistema di pubblicazione per Blog. Lo scarno comunicato che annunciava l'acquisizione, parlava in modo molto vago di sinergie e future opportunità che nasceranno grazie a questa unione: ma cosa ha spinto in realtà Google a questo passo? Una risposta potrebbe essere proprio la possibilità di migliorare il suo servizio di News, che potrà così beneficiare della maggiore velocità assicurata dai Weblog rispetto ai siti di news tradizionali. In realtà, l'archivio di Blogger è già a disposizione dei crawler di Google, ma l'integrazione potrebbe portare ad una indicizzazione in tempo reale dei messaggi.

www.blogger.com
www.google.com

SDS200 oscilloscopio digitale per PC

Un normale PC diventa una stazione di test e misure elettroniche

Giakova presenta sul mercato italiano SDS200, uno strumento che trasforma un normale PC in un oscilloscopio digitale completo di analizzatore di spettro. Utilizzando una sofisticata tecnologia di conversione analogica/digitale SDS200 coniuga i tradizionali benefici di un DSO (oscilloscopio digitale) da 200 MHz e campionamento in tempo equivalente a 5 GS/sec con i vantaggi dell'integrazione nella piattaforma PC. L'interfaccia USB consente la connessione dello strumento senza alcuna necessità di installare elementi hardware aggiuntivi nel PC; la configurazione è semplice, lo strumento può essere connesso a PC acceso e l'alimentazione viene prelevata dall'interfaccia, senza necessità di alimentatori addizionali o batterie. Si possono così sfruttare appieno le caratteristiche del PC, come monitor a colori, disco rigido, stampanti, trasferimento dei segnali acquisiti nei fogli di Excel e Word, in formato bitmap o in formato numerico senza interfacce e software addizionali. Lo strumento viene fornito completo di due sonde con attenuazione selezionabile x1 e x10 da 100 MHz, cavo USB, CD-Rom, manuale d'installazione. Il tutto è inserito in una borsa morbida di dimensioni contenute.

www.giakova.com

Bluetooth USB Adapter

Sitecom presenta un prodotto per dotare il PC di connettività BlueTooth, utile per interfacciare telefonini, palmari e SmartPhone al proprio Personal Computer

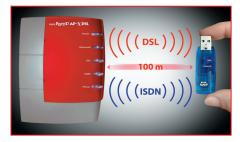
Sitecom, azienda leader del mercato nel settore informatico dei prodotti per la connessione, ha di recente presentato il prodotto Bluetooth USB Adapter. Grazie a questo praticissimo accessorio (una piccolissima chiave da inserire in uno slot USB) si disporrà infatti di uno strumento intelligente che consente di scambiare dati con dispositivi dotati di tecnologia Bluetooth. Sarà

per esempio possibile scambiare informazioni con telefoni cellulari, PDA, SmartPhone; utile in tutte quelle situazioni in cui l'ingombro dei cavi diventa un vero tormentone. Il dispositivo può anche essere adottato per creare una mini-rete fra PC, per esempio per scambiare dati tra il PC portatile e il personal computer, o ancora per scambiare dati tra due postazioni PC distanti tra loro anche fino a circa 10 mt. In tutte queste situazioni, e in molte altre ancora, il prodotto Bluetooth USB adapter apporterà notevoli vantaggi e una sicura flessibilità di lavoro. Il prodotto è disponibile a ¤ 69,95 IVA (inclusa).

www.sitecom.com

ISDN, ADSL e Bluetooth insieme

AVM risolve il problema di chi vuole condividere un'unica connessione ad Internet tra più utenti

Al CeBIT 2003 AVM ha presentato un prodotto piuttosto interessante per tutti coloro che desiderano sfruttare un collegamento ADSL da condividere tra tutti i computer dell'ufficio, senza però installare quegli antiestetici e fastidiosi collegamenti cablati. BLUE!Fritz ADSL, difatti, unisce in un unico prodotto funzioni di controller ADSL, adattatore ISDN ed access point con connessione Bluetooth. Si può collegare direttamente alla borchia ISDN, o allo splitter ADSL, inoltre possono essere connessi due dispositivi come telefoni o fax analogici. Utilizzando la tecnologia Bluetooth, inoltre, è possibile collegarsi all'access point ed avere accesso alle risorse Internet senza la necessità di stendere cavi e bucare muri o pavimenti, semplicemente utilizzando uno dei tanti adattatori wireless Bluetooth disponibili sul mercato. A questo proposito, AVM propone Blue!FRITZ USB, un controller bluetooth per notebook o PC che consente di sfruttare la connessione ISDN/ADSL via bluetooth. Oltre ad essere piccolo e leggero, ogni adattatore Bluetooth viene configurato con una chiave d'accesso modificabile dall'utente per garantire che solo gli utenti Bluetooth autorizzati abbiano

accesso al collegamento. Ottima la garanzia, che copre i dispositivi per 5 anni.

www.avm.com

Zelig-One: una piuma da 128 MB

Peso e dimensioni ridottissime per la chiave USB "da portafogli" di Hamlet

Pesa solo tre grammi ed ha dimensioni ridottissime (2,8 mm di spessore, lunghezza inferiore ai 40 mm e larghezza inferiore ai 20 mm) il nuovo Zelig-One di Hamlet, un drive USB che si annuncia come il più piccolo al mondo. Grazie alla sua estrema leggerezza è possibile memorizzare e trasferire i dati da un computer all'altro senza alcun problema, anche perché il drive può essere trasportato anche nel portafogli o in un organizer. Lo Zelig-One è compatibile con tutti i sistemi operativi Windows (solo con Windows 98 occorre installare i driver), nonché con i sistemi Macintosh con Mac OS 9.X o Linux Kernel 2.4.0 e versioni successive. Hamlet Zelig-One è disponibile in tre versioni: da 32, 64 e 128MB, con prezzi di 29, 45 e 75 Euro rispettivamente.

www.avm.com

dBASE plus

Una soluzione potente e flessibile per gli amministratori di basi di dati

SCHEDA TECNICA

Nome prodotto:
dBASE PLUS
Produttore:
dBASE inc.
Sito ufficiale:
www.dbase.com
Distributore Italiano:
Ecosoft Srl
www.db2k.it
Prezzo € 680

Che siate un professionista dell'informatica o che vi troviate a muovere i primi passi nel campo dei database, dBASE può rappresentare un'ottima soluzione per le vostre esigenze. dBASE Plus è una ambiente RAD (Rapid Application Development) per la creazioni di potenti applicazioni data-driven e applicazioni Web grazie a numerosi tool per l'amministrazione di database, un avanzato modello di programmazione object-oriented e l'opportunità di collegarsi ad un ampio ventaglio di soluzioni legacy. All'interno del pacchetto è incluso il Borland Database Engine (BDE), che permette una facile connettività verso le tabelle in formato dBASE (incluso il nuovo DBF7), oltre al supporto nativo per tutti più diffusi DBMS. Attraverso una interfaccia decisamente innovativa e orientata all'utente finale, è possibile interagire con l'applicativo con delle semplici azioni di drag&drop e, molto spesso, grazie ai numerosi Wizard presenti, si è in pratica "guidati" verso la realizzazione della sistema informativo che si va a implementare. Gli utenti meno smaliziati apprezzeranno sicuramente il *dQuery/Web*: una soluzione in grado di generare applicazioni stand alone e applicazioni Web a partire da una base di dati, con pochi semplici passaggi. Per gli amministratori professionisti, dBASE plus offre l'impagabile capacità di accedere a tutti i più diffusi formati di database: Access, SQL Server, Oracle, Paradox e DB2. Davvero interessante la possibilità di leggere e scrivere dati e tabelle senza la necessita di effettuare conversioni esplicite da un formato all'altro. L'interfaccia di *dQuery/Web* consente di tenere sotto controllo allo stesso tempo la struttura dei dati ed il contenuto vero e proprio del DB: la porzione superiore dello schermo è occupata dal *dataModule*, in cui è possibile trascinare (con semplici operazioni di drag&drop) i vari elementi che compongono la struttura del nostro sistema informativo. In basso, abbiamo invece l'opportunità di visualizzare, modificare e aggiungere dati al database, effettuare Query attraverso la cosiddetta Custom View e generare automaticamente dei report a partire dalle viste personalizzate. A partire dal *dataModule* (l'insieme di modelli e dati che compongono il nostro database), è possibile generare rapidamente una serie di pagine di accesso al DB: con la funzione One-Click Web saremo pronti per pubblicare le pagine di accesso ai dati in un batter d'occhio! Nel complesso l'interfaccia si presenta decisamente piacevole, anche se non particolarmente intuitiva, per via di alcune scelte controcorrente. L'utilizzatore abituale avrà sicuramente modo di ambientarsi ma la prima impressione è di un certo disorientamento. In compenso, in numerosi wizard consentono di raggiungere subito degli ottimi livelli di produttività. In conclusione, dBASE plus rappresenta una ottima soluzione soprattutto per gli amministratori di database che abbiano a che fare con sistemi eterogenei, oltre che una importante occasione per i vecchi utenti di dBASE per far convergere le proprie conoscenze e le proprie possibilità in direzione di Internet e di applicazioni data-driven.

Fig. 2: **Dati e strutture a portata di clic.**

Fig. 1: **Notare i numerosi wizard disponibili.**

Il Software di ioProgrammo

DirectX9

VERTEX BLENDING E PER-PIXEL LIGHING

Questo mese tratteremo due argomenti di grande utilità per qualunque tipologia di gioco. Per i beginners parleremo di semplice vertex blending, che è alla base di tecniche più complesse come ad esempio il mesh skinning. Per gli esperti faremo un'introduzione al per-pixel lighing con le mappe di attenuazione. L'HLSL mostrerà finalmente il suo potenziale e la sua semplicità.

VERTEX BLENDING E PRIMITIVE GRAFICHE

Possiamo suddividere le mesh in "statiche" e "animate". A loro volta le mesh animate possono essere "rigide" o "deformabili". Quando una delle proprietà della mesh (ad esempio, posizione, dimensione etc) cambia nel tempo, diremo che la mesh è animata. Le mesh "rigide" vengono animate nel tempo semplicemente applicando ad esse una matrice. Tale matrice prende ad uno ad uno tutti i vertici e li trasforma, spostandoli e ruotandoli. Per disegnare un cubo che ruota ad esempio, basta semplicemente alterare la matrice WORLD con una rotazione ad angolo crescente nel tempo. Alle mesh rigide quindi si possono applicare solo trasformazioni "standard", quindi traslazioni, rotazioni, scale uniformi, proiezioni ecc. Fin qui nulla di nuovo, senza nemmeno saperlo abbiamo usato questa tecnica negli esempi del precedente numero della nostra rubrica. Sulle mesh "deformabili" invece, si possono eseguire una serie di trasformazioni più complesse. Queste mesh si basano sul meccanismo del "*vertex blending*". Il vertex blending è una tecnica che permette ad ogni singolo vertice della mesh di essere "influenzato" da più trasformazioni. Ho detto "influenzato" e non "trasformato" perché ogni matrice di trasformazione contribuisce solo in percentuale al risultato finale. Ogni vertice deve quindi portare con sé le informazioni di "blending" (detti anche "pesi" in gergo), cioè le percentuali di influenza di ogni singola matrice su di esso. E' facile capire l'utilità di questa tecnica: un personaggio animato quale ad esempio un umanoide con il vertex blending può assegnare delle zone di influenza alle varie "ossa" dello scheletro che lo compone. Un gomito ad esempio è influenzato dalla matrice di trasformazione del braccio, ma anche da quella dell'avambraccio. Combinando tra di loro queste matrici in maniera gerarchica si può modellare lo scheletro di un "character", il quale, una volta applicati i giusti pesi ai vertici, deformerà la mesh. Cerchiamo di capire come implementare questa tecnica in DirectX9. Come sempre nel CD allegato a questo numero troverete un esempio molto semplice, commentato in ogni sua riga, del quale riportiamo solo i punti salienti. Il nostro esempio, chiamato "*vertex blending*" tanto per essere originali, legge una mesh dal disco attraverso la funzione *D3DXLoadMeshFromX* e assegna "via codice" i pesi di blending di quelle che da ora in poi chiameremo le "ossa" del blending. Il termine "ossa" è preso in presto alla computer grafica per la tecnica del character skinning, usata per i personaggi animati. Nel 99% dei casi, nella vita reale questi pesi saranno generati da un qualche programma di animazione 3D come ad esempio 3DStudioMax (con i plugin *Biped* e *physique*), Maya, Lighwave ecc. Sul sito microsoft sono disponibili degli "extras" del DXSDK9 che contengono plugin di esportazione in formato *.x* per i maggiori pacchetti di grafica sul mercato. Tali plug-in vengono forniti con tanto di sorgenti e quindi potete anche adattarli per le vostre esigenze specifiche. Tornando al nostro esempio, una volta letta la mesh con *D3DXLoadMeshFromX*, ed

Fig. 1: Le bones in azione, un gomito in cui movimento e subordinato al movimento del braccio e dell'avanbraccio (in 3DStudioMax).

aver letto le texture ad essa assegnate (come potete vedere nella banalissima funzione *LoadMesh* dell'esempio) viene effettuato un "clonaggio" della texture, cambiandone il FVF. Spieghiamo meglio quest'ultimo punto. Avendo letto la mesh dal disco sicuramente ogni vertice conterrà i seguenti campi:

```
struct MESHVERTEX
{
    D3DXVECTOR3 posizione;
    D3DXVECTOR3 normale;
    DWORD diffuse;
    D3DXVECTOR2 texture;
};
```

Esso contiene la posizione del vertice, la normale ad esso (usata per l'illuminazione) il colore diffuso del vertice e un vettore 2D contenente le coordinate *texture*. Come ben vedete non c'è spazio per le informazioni sui pesi delle singole ossa di cui abbiamo parlato in precedenza. Ovviamo a questo problema creando una nuova definizione per i vertici della mesh (CUSTOMVERTEX) e inseriamo un valore float (subito dopo la posizione) che rappresenta il peso di blending. In *CreateDeviceObjects* "cloniamo" letteralmente la mesh usando questa nuova definizione di vertice:

```
CUSTOMVERTEX
{  D3DXVECTOR3 posizione;   //Posizione
   float blend; //Peso di blending
   D3DXVECTOR3 normale;    //Normale
   D3DXVECTOR2 texture; //Coordinate Texture };
const DWORD CUSTOMVERTEX_FVF =
          D3DFVF_XYZB1 | D3DFVF_NORMAL |
                              D3DFVF_TEX1;
D3DXMATRIX matWorld1,matWorld2;
bool CreateDeviceObjects()
{ //..omissis
  D3DXMesh_Ptr tmpMesh;
  LoadMesh(_T("ASD.X"),tmpMesh,g_materials,
                               g_textures)
  hr = tmpMesh->CloneMeshFVF(
                      D3DXMESH_MANAGED,
   CUSTOMVERTEX_FVF, g_device,&g_mesh);
  AssignBlendWeights();
  //..omissis }
void AssignBlendWeights()
{ HRESULT hr;
  CUSTOMVERTEX* v=0;
  hr = g_mesh->LockVertexBuffer(
        D3DLOCK_NOSYSLOCK,(void**)&v);
  if(FAILED(hr)) return;
  D3DXVECTOR3 bmin,bmax;
  D3DXComputeBoundingBox((D3DXVECTOR3*)v,
           g_mesh->GetNumVertices(),sizeof(
                CUSTOMVERTEX),&bmin,&bmax);
  float xsize = bmax.x - bmin.x;
  for(int i=0; i<g_mesh->GetNumVertices();
                                      i++)
     v[i].blend = sinf(D3DX_PI*
              (v[i].posizione.x-bmin.x)/xsize);
  g_mesh->UnlockVertexBuffer(); }
```

La *ID3DXMesh::CloneMeshFVF* crea una mesh identica alla precedente, cambiandone la definizione del vertice e copiando tutte le informazioni presenti in entrambi i formati di vertice (la posizione, la normale e le coordinate texture in questo esempio). Successivamente chiamiamo la *AssignBlendWeights* per impostare i pesi via codice. Per fare questo eseguiamo un "lock" sui vertici della mesh e assegniamo il peso di blending attraverso una funzione sinusoidale basata sulla coordinata X del vertice. Ogni qualvolta avete bisogno di accedere in lettura o scrittura ai dati di una mesh, dovete farne il "lock" come mostrato nel listato. Quando avete finito di leggere o scrivere i dati nella mesh dovete chiamare la relativa procedura di Unlock. Dimenticare quest'ultimo passaggio molto probabilmente farà fallire il disegno della mesh e vedrete un bel nero a schermo. Come diceva un famoso programmatore 3D:

"The tricky part in 3Dcoding is that there are too many ways to code a black screen"...

Una volta fatto questo la procedura di disegno è molto semplice:

```
void Draw()
{ //..omissis
  //Impostiamo le 2 bones
  g_device->SetTransform(D3DTS_WORLD,
                             &matWorld1);
  g_device->SetTransform(D3DTS_WORLD1,
                             &matWorld2);
  //Attiviamo il vertex blending a 2 bones
  g_device->SetRenderState(
   D3DRS_VERTEXBLEND, D3DVBF_1WEIGHTS);
  for(UINT i = 0; i < g_materials.size(); i++)
  { //Impostiamo il materiale del subset
    g_device->SetMaterial(
                 &g_materials[i].MatD3D);
    g_device->SetTexture(0,g_textures[i]);
    g_mesh->DrawSubset(i); }
  //..omissis }
```

Riempite la *matWorld1* e la *matWorld2* con delle trasformazioni di nostro gradimento le passiamo a *DirectX* con la *SetTransform*, specificando per la prima *D3DTS_WORLD* e per la seconda *D3DTS_WOLRD1*. Poi abilitiamo il vertex blending impostando il renderstate *D3DRS_VERTEXBLEND* e "indicando" a DirectX che useremo 1 solo peso di blending. Infine disegniamo i vari "subset" della mesh con le relative texture. Probabilmente molti di voi si staranno chiedendo come si possano usare due "ossa", avendo specificato solo un peso di blending nella definizione del vertice. Il motivo sta nel fatto che l'ultimo peso di blending viene calcolato con la formula

peso_ultimo_vertice = 1.0f – (somma_pesi_altri_vertici).

Il significato di questa formula è semplicissimo, dato che la somma dei vari pesi deve dare il 100% (1.0f), l'ultimo peso può essere calcolato per differenza tra 1.0f e la somma degli altri; DirectX automatizza questa operazione.

PRIMITIVE GRAFICHE

Le mesh di D3DX sono sostanzialmente dei wrapper per i vertex e index buffer. Un *vertex buffer* è, come suggerisce il nome stesso, un array di vertici. Un *index buffer* è un array di indici. Gli elementi dell'index buffer "indicizzano" nel vertex buffer (che bel gioco di parole), per indicare a DirectX i triangoli che compongono la mesh. Questi due buffer lavorano in tandem per intenderci. In realtà potremmo usare direttamente index e vertex buffer, ma la *ID3DXMesh* ci semplifica molto la vita. Non tutto però si può fare con la *ID3DXMesh*, quindi a volte siamo costretti a lavorare direttamente con le cosiddette "primitive grafiche". Possiamo specificare a DirectX i triangoli da disegnare attraverso uno dei valori dell'enumerazione *D3DPRIMITIVETYPE*. Esistono tre modalità per disegnare i triangoli:

1) **Triangle list:** questa è la più semplice, ogni gruppo di tre indici dell'index buffer forma un triangolo. Facendo un esempio:

```
WORD i_list[]= {0,1,4,    //Triangolo 1
                1,3,4,    //Triangolo 2
                1,2,3};  //Triangolo 3
```

2) **Triangle strip:** ogni indice aggiunto nell'index buffer crea un nuovo triangolo, gli altri due indici necessari sono gli ultimi due, immediatamente precedenti al corrente:

```
WORD i_strip[]={0,1,4, //Triangolo 1
                    (formato da 0,1,4)
    2,    //Triangolo 2 (formato da 1,4,2)
    3};  //Triangolo 3 (formato da 4,2,3)
```

3) **Triangle fan:** ogni indice aggiunto nell'index buffer crea un nuovo triangolo, gli altri due necessari sono il primo della lista e l'ultimo immediatamente precedente al corrente:

```
WORD i_fan[]= {1,2,3, //Triangolo 1
                    (formato da 1,2,3)
      4,  //Triangolo 2 (formato da 1,3,4)
    0};  //Triangolo 3 (formato da 1,4,0)
```

Nel CD trovate un semplicissimo esempio di nome "*Primitive*" che disegna 3 triangoli disposti a formare un quadrato sullo schermo usando le varie primitive (cambiabili con la barra spazio). Come potete immaginare, la differenza tra questi metodi di disegno sta nelle performance. È palese che la più comoda sia la *triangle list*, ma è anche quella che occupa più memoria. Se *N* è il numero di triangoli da disegnare, la triangle list usa *N*3* indici, la *triangle strip* e la *triangle fan* ne usano *N*+2. Allo stesso tempo però può non essere sempre facile convertire una *mesh* in *fan* o *strip*. Convertire efficientemente una mesh arbitraria in strips non è roba da poco, ma per fortuna ci sono delle funzioni di D3DX che automatizzano l'operazione: *D3DXConvertMeshSubsetToSingleStrip* e *D3DXConvertMeshSubsetToStrips*. Se vi interessa questo argomento di ottimizzazione, l'esempio "*OptimizedMesh*" del DXSDK è quello che fa per voi. La ID3DXMesh usa SEMPRE *Triangle List* indicizzate (cioè con index buffer).

PER PIXEL LIGHING

In una delle trattazioni precedenti abbiamo usato l'HLSL per proiettare delle texture sui poligoni. Non ci siamo però dilungati sul funzionamento degli Effect Files e dell'HLSL per motivi di spazio, ma oggi cercheremo di rimediare. Partiamo dal codice:

```
float4x4 matWorldViewProj;
float4x4 matWorld;
float4 light;
texture diffuseTexture;
texture att1DTexture;
texture att2DTexture;
float4 ambient = {0.1,0.1,0.1,1};
struct VS_OUTPUT
{
```

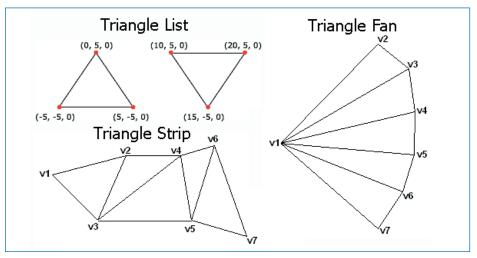
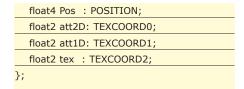



Fig. 2: **I tre tipi di primitive di Direct3D: Triangle List, Triangle Strip e Triangle Fan (immagini tratte dal DXSDK).**

```
    float4 Pos   : POSITION;
    float2 att2D: TEXCOORD0;
    float2 att1D: TEXCOORD1;
    float2 tex   : TEXCOORD2;
};
```

La struttura di un "*Effect File*" è di solito simile a quella che vedete nel codice sopra riportato. All'inizio del file si dichiarano delle variabili che vengono settate dall'applicazione chiamante, attraverso l'interfaccia *ID3DXEffect* e i metodi *Set/Get*. Tra di esse ci sono quasi sempre matrici di trasformazione world, view e space, da passare al vertex shader. In generale qualunque parametro controllato dall'applicazione è dichiarato in questa sezione. Come si può notare, oltre a dichiarare queste variabili si può anche inizializzarle con dei valori predefiniti, come nel nostro caso il parametro "*ambient*".
Un'altra cosa che si nota è che ogni variabile ha un suo "tipo", esattamente come un qualsiasi linguaggio di programmazione. I nomi sono abbastanza esplicativi, *float4x4* rappresenta una matrice 4x4, *float4* un vettore a 4 componenti e *texture* un handle ad 1 texture. Ovviamente esistono altri tipi predefiniti e potete leggerne la lista nella pagina dedicata all'HLSL sulla documentazione dell'SDK. Tra le dichiarazioni spicca una struct di nome *VS_OUTPUT*. In essa vengono dichiarati i parametri che il vertex shader scrive in output. La notazione dei due punti (:) seguita da una parola chiave viene chiamata sematic, indica l'uso che si farà nello shader di quel campo. Queste semantic comprendono *POSITION* per la posizione *TEXCOORDn* per i vari stage di coordinate texture, *COLOR* per i colori, *NORMAL* per le normali e così via.

```
VS_OUTPUT VS(float4 Pos : POSITION, float2
                    diffTexCoord : TEXCOORD0)
{
    //Omissis
}
float4 PS( float2 att2D : TEXCOORD0,
                    float2 att1D : TEXCOORD1,
               float2 tex : TEXCOORD2) : COLOR
{
    //OMISSIS...
}
float4 PSEXP(float2 att2D : TEXCOORD0,
float2 att1D : TEXCOORD1,
float2 tex : TEXCOORD2) : COLOR
{
   //omissis...
}
```

Successivamente vengono dichiarate alcune funzioni. In questo caso, una è il vertex shader e le altre due sono pixel shader. Come fa ben capire il nome, il vertex shader è una "funzione" che lavora su ogni singolo vertice dei triangoli che vogliamo disegnare; riceve in input dei parametri che si trovano nella dichiarazione del vertice e non può in nessun modo accedere ad altri vertici al di fuori di quello corrente. Un vertex shader accetta un vertice in input e ritorna un vertice trasformato in clip space, con ornamenti vari quali coordinate texture e colori. Si può però accedere ad altri parametri "globali", settati dall'applicazione, come ad esempio quelli dichiarati nella sezione iniziale dell'effect file (nel nostro caso *matWorld*, *matWorldViewProj* ecc.). In realtà c'è ancora un'altra famiglia di parametri, i parametri "uniform", molto utili per compilare diverse versioni dello stesso shader, ma questo sarà argomento dei prossimi numeri. Anche i para-

metri del vertex shader sono dichiarati con la semantic che specifica il loro uso. Lo stesso vale per il pixel shader. Ma come sono legati vertex e pixel shader? Dopo aver eseguito il vertex shader su tutti i vertici della mesh, nella rasterizzazione di ogni triangolo, DirectX interpola i valori dei singoli vertici. Poi per ogni pixel sullo schermo (in realtà è più corretto il termine *"fragment"*) chiama la funzione *Pixel Shader* che riceve in input tali parametri "interpolati". Come spigheremo in seguito, l'interpolazione dipende dalle impostazioni del sampler. Giusto per fare un esempio pratico, se disegnate due triangoli disposti a formare un quadrato e nel vertex shader date delle coordinate tra 0 ed 1 per mappare una texture quadrata su di esso, nel punto (0.5,0.5) al centro del quadrato, il pixel shader riceverà esattamente la coordinata texture (0.5, 0.5).

```
technique TecnicaDST2
{ pass P0
  { VertexShader = compile vs_1_1 VS();
    PixelShader  = compile ps_1_1 PS(); }
}
technique TecnicaEXP2
{pass P0
  { VertexShader = compile vs_1_1 VS();
   PixelShader  = compile ps_1_1 PSEXP(); }
}
```

Dopo aver definito tutte le variabili e le funzioni è il momento delle "tecniche". Una "tecnica" è composta da 1 insieme di pass. Fare più pass in una stessa tecnica significa disegnare più volte lo stesso triangolo sullo schermo, in genere abilitando delle particolari opzioni di blending in modo da ottenere l'effetto desiderato. Si possono avere anche più tecniche nello stesso file, ognuna contraddistinta da un nome univoco. Il fatto di avere diverse tecniche permette di compilare diverse versioni dello stesso effetto, o se volete diversi effetti nello stesso file *effect*. Ad esempio si potrebbero creare tecniche diversificate secondo l'acceleratore video di cui si dispone e sfruttare al massimo le sue caratteristiche. All'interno di ogni pass si possono impostare una miriade di "renderstate" di DirectX (gli stessi che si impostano con *IDirect3DDevice9::SetRenderState*), elencati nella pagina relativa agli effect file nella documentazione del DXSDK. In questo esempio impostiamo semplicemente *Vertex* e *Pixel shader* attraverso il comando *"compile"*, che appunto compila la nostra funzione HLSL in linguaggio assembler. La versione di questo assembler viene specificata tramite l'istruzione *vs_1_1* o *ps_1_1*. Cambiando la versione dell'assembler ovviamente abbiamo a disposizione più istruzioni all'interno del nostro script HLSL. I pixel shader 2.0 ad esempio sono enormemente più potenti in termini di numero, versatilità e precisione rispetto ai precedenti, al momento sono supportati in hardware solo dalla serie Radeon 9x00 della ATI e dalle imminenti GeForceFX. Le schede ATI radon 8x00 hanno il supporto pixel shader 1.4 e le GeForce 3 e 4 hanno supporto pixel shader 1.1. In realtà le GeForce4 (versioni NON MX) supportano anche una versione chiamata pixel shader 1.3 che è quasi del tutto simile alla 1.1, ma con qualche istruzione in più. I numeri di versione non sono particolarmente indicativi, giacché i pixel shader 1.4 sono enormemente più potenti dei pixelshader 1.1 ed 1.3. Dopo questa introduzione agli effect file e all'HLSL passiamo all'argomento advanced di questo mese: l'attenuazione "per pixel". Il lettore medio di questa sezione avrà sicuramente giocato con le luci di DirectX almeno una volta nella sua vita (*SetLight*, *SetMaterial* ecc). La facilità d'uso di questo sistema si scontra però con i suoi grandi limiti. Esso infatti calcola l'illuminazione su ogni singolo vertice e poi effettua delle interpolazioni tra i vertici di ogni triangolo. Come sicuramente avrete notato i risultati non sono dei più soddisfacenti, soprattutto per mesh con basso poligonaggio. Per aumentare la qualità di questo sistema bisognerebbe aumentare in maniera spropositata il numero di triangoli, in modo da infittire la rete di vertici sui quali viene calcolata l'illuminazione. Esistono delle alternative. Immaginate di avere una scena con una luce puntiforme. Per chi non se lo ricordasse, una luce puntiforme emana raggi di luce in tutte le direzioni. I parametri caratteriz-

Fig. 3: L'esempio "per pixel lighing" in azione nelle sue due modalità, DST(D^2) e EXP(D^2).

zanti questa luce sono la sua posizione nel mondo (world space) e il suo raggio d'influenza. Abbiamo deciso di illuminare ogni singolo pixel della scena con la formula: *Intensità = 1 - Attenuazione*. La formula per il calcolo dell'attenuazione è data da: *Attenuazione = D2/R2* in cui *D2* è la distanza del pixel dalla luce ed *R2* è il quadrato del raggio della luce.
Analizzando questa semplicissima formula ci accorgiamo che man mano che la luce e il pixel si allontanano, la *D* aumenta, il termine *Attenuazione* aumenta e quindi l'intensità (per come è definita) diminuisce. La nostra *D2* è una semplice formula di distanza nello spazio quindi *D2 = x2 +y2 +z2*.
Riarrangiando abbiamo:

Intensità = 1 – (x/R)2 +(y/R)2 +(z/R)2

Ci rimane da trovare un modo per computare i termini quadratici di questa equazione. Niente di più semplice: salviamo nei pixel di una texture *2D* la funzione:

f(x,y) = (x/R)2 +(y/R)2

che graficamente è rappresentata da un cerchio che si sfuma dal centro verso l'esterno. In una texture *1D* salviamo la funzione: *g(z) = (z/R)2*. Una texture *1D* non è niente di magico, è una texture rettangolare di altezza 1. Per recuperare l'attenuazione basta fare:

Attenuazione = TEXTURE2D(x,y) + TEXTURE1D(z).

I valori da mandare in pasto a queste texture ovviamente sono le distanze normalizzate (cioè divise per *R*) dal pixel alla luce e quindi:

x0 = (xp - lightX)/R
y0 = (yp - lightY)/R
z0 = (zp - lightZ)/R

Ovviamente bisogna fare gli opportuni cambiamenti di riferimento: le coordinate texture sono da *[0,0], [0,1], [1,1]* e *[1,0]* mentre a noi serve avere *[0,0]* al centro della texture. Quindi scaliamo la texture di un fattore 2 e la trasliamo di 0.5:

s = x0/2 + 0.5
t = y0/2 + 0.5
r = z0/2 + 0.5

s e *t* sono le coordinate texture della mappa di attenuazione *2D*, mentre la *r* è la coordinata della texture *1D*.

```
int size = 64;
D3DLOCKED_RECT r_dst2,r_exp;
hr = att2D->LockRect(0,&r_dst2,0,
                              D3DLOCK_NOSYSLOCK);
if(FAILED(hr))
   return false;
hr = att2DExp->LockRect(0,&r_exp,0,
                              D3DLOCK_NOSYSLOCK);
if(FAILED(hr))
   return false;
float scale = 1.8f;
for(int y = 0; y < size; y++)
{   float fy = (2*y+1)/(float)size-1;
    float efy = fy * scale;
    efy = exp(-(efy*efy));
    for(int i = 0; i < size; i++)
    {  BYTE* D2 = (BYTE*)r_dst2.pBits+
                              r_dst2.Pitch*i+y;
       BYTE* Exp = (BYTE*)r_exp.pBits +
                              r_exp.Pitch*i +y;
       float fi = (2*i+1)/(float)size-1;
       float efi = fi * scale;
       efi = exp(-efi*efi);
       *Exp = efi*efy*255;
       float dst2 = fi * fi + fy * fy;
       if(dst2 > 1)
          dst2 = 1;
          *D2 = dst2*255; }
}
att2D->UnlockRect(0);
att2DExp->UnlockRect(0);
```

Con questo codice creiamo la mappa d'attenuazione, ed un'altra esponenziale, che ha una formula simile a quella quadratica. Gli shaders non fanno altro che implementare le formule descritte in precedenza. In realtà si è fatta una piccola ottimizzazione, al posto di usare due texture (una *2D* ed un *1D*) si usa solo la texture *2D*. LA texture *1D* è "emulata" mettendosi sulla riga centrale della texture *2D (att1D.y = 0.5f).*

```
VS_OUTPUT VS(float4 Pos : POSITION,
         float2 diffTexCoord : TEXCOORD0)
{ VS_OUTPUT Out = (VS_OUTPUT)0;
  light.w *= 2;
  float3 WorldPos = mul(Pos,matWorld);
  float3 light_to_vertex = WorldPos-light;
  float3 att_texcoord = light_to_vertex
                              /light.w+0.5;
  Out.Pos = mul(Pos, matWorldViewProj);
  Out.att2D = att_texcoord.xy;
  Out.att1D.x = att_texcoord.z;
  Out.att1D.y = 0.5f;
  Out.tex = diffTexCoord;
  return Out; }
float4 PS( float2 att2D : TEXCOORD0,
float2 att1D : TEXCOORD1,
float2 tex : TEXCOORD2) : COLOR
{  float4 spot = (saturate(1-tex2D(
                          mySampler0,att2D)-
   tex2D(mySampler1,att1D))+ambient);
   return tex2D(mySampler2,tex)*spot*4;}
```

L'unica cosa da menzionare è che usiamo la saturate nel pixel shader per evitare che leggendo le mappe di attenuazione si possano avere valori fuori dal range *[0,1]*.
Un ringraziamento speciale a Luciano Iurino dei PM Studios di Bari (*http://www.pmstudios.it*) per aver concesso l'utilizzo di un suo modello dal gioco in lavorazione ETROM (*http:// www.etrom.net*) per l'esempio sul *per pixel lighing*.
Se l'argomento vi appassiona probabilmente passerete i prossimi mesi a spulciare tutto il sito della NVidia e della ATI. ASD, Buon coding!

Stefano Cristiano

DirectX

DirectX oltre a passare triangoli come Fan, List e Strips può gestire anche superfici parametriche curve (vari tipi di patch), se supportate in hardware.

Sul Web

Interessante sito con Tutorials DirectX e non solo
http://www.dotnethell.it **(sezione DirectX)**

Un'alternativa all'esportatore da 3DstudioMax a .X dell'SDK. Fixa alcuni bug ma ne mette altri
http://www.pandasoft.demon.co.uk/directxmax4.htm

Tutorials avanzati di illuminazione con schede di fascia GeForce 1/2
http://www.ronfrazier.net/apparition/index.html

Una simpatica community di newbies e non solo tra i quali trovare aiuto nel arduo cammino del 3Dcoding
http://www.mutantpenguins.net/forum/

Il sito personale dell'autore dell'articolo, pieno di materiale per spunti e approfondimenti
http://pagghiu.gameprog.it

Sul sito NVidia ci sono papers per implementare tecniche che espandono i concetti esposti nell'articolo
http://developer.nvidia.com

Sul sito ATI ci sono dei nuovi papers da non perdere tra i quali "Performance Optimization Techniques for ATI Graphics Hardware with DirectX® 9.0"
http://www.ati.com/developer

Tutorials DirectX9
http://www.booyah.com/articles-dx9.html

BLOBJ

Uno strumento C.A.S.E. (Computer-Aided Software Engineering) di supporto alla produzione industriale di software

L'ambiente di sviluppo consente di generare codice Java senza una conoscenza specifica del linguaggio. Blobj permette di ridurre in maniera drastica i tempi di sviluppo del codice di routine, consentendo al programmatore di concentrarsi maggiormente sulla logica applicativa e la produzione di funzionalità complesse.

L'AMBIENTE

Dal Menu principale di Blobj è possibile accedere a tutte le funzioni ed ai comandi del software. Inoltre, una serie di pannelli permettono di visualizzare i diversi elementi nella fase di progettazione. Sopra i pannelli, una barra di strumenti propone tasti per l'esecuzione rapida di alcuni dei comandi principali di Blobj, oltre al controllo della rappresentazione grafica dell'applicazione corrente. Col tasto destro del mouse, poi, si può accedere ad importanti funzionalità tramite menu contestuali. La schermata principale si presenta come in Fig. 1.

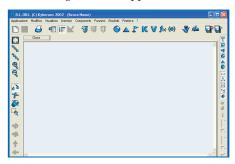



Fig. 1: **Schermata principale.**

I COMPONENTI

Sulla barra degli strumenti sono presenti alcuni pulsanti di fondamentale importanza:

Questo tasto visualizza la finestra di dialogo "Classe" per la creazione di una nuova classe del progetto corrente. Esempi di classi possono essere il gatto, il cane, la mucca… essi sono caratterizzati dal colore, dal tipo e dal verso. Le caratteristiche appena elencate sono "attributi " delle classi e ci consentono di differenziarle l'una dall'altra.

In Blobj gli attributi possono essere creati dal tasto rappresentato.

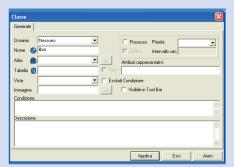
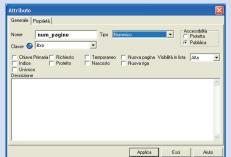
Ogni classe è caratterizzata dagli attributi e da un comportamento, quest'ultimo può essere impostato dal tasto rappresentato.

SVILUPPIAMO UN'APPLICAZIONE

Un libro può essere visto come una classe di oggetti aventi caratteristiche comuni: tutti i libri contengono delle pagine che possono essere sfogliate o lette. All'interno di una classe vengono dichiarati i metodi e gli attributi. Nel nostro esempio un metodo potrebbe essere "*giraLaPagina*" ed un attributo "*numero_pagine*". In tal modo abbiamo utilizzato il termine "libro" per generalizzare un concetto relativo a qualcosa che contiene pagine da sfogliare, leggere, strappare… ossia ci riferiamo ad un insieme di oggetti con attributi comuni. Realizziamo assieme la classe libro con l'attributo '*num_pagine*'. Utilizzando la connessione ODBC,è possibile importare una struttura dati esistente. Nell'operazione di importazione Blobj genera una classe per ogni tabella. Dopo aver eseguito i passi 1 e 2 si seleziona la voce *Database | Importa Struttura dal menu Funzioni,* si sceglie la fonte dei dati e si clicca su Applica. Dal menu contestuale della classe appena creata, possiamo accedere alle sue proprietà e notare che i campi della tabella che abbiamo importato, sono diventati gli attributi della nostra classe.

SCHEDA TECNICA
Blobj non necessita di particolare Hardware. È necessaria una porta parallela per la chiave d'attivazione del prodotto. Per connessioni ODBC, può essere necessaria una scheda di rete.

Costruiamo la classe Libro

1 **Dal menu *Applicazione* selezionare la voce *Proprietà*, digitare nel campo nome '*testi*'; selezionare un database da una delle fonti ODBC configurate sulla macchina o cliccare sul tasto… per crearne una nuova e premere *Applica.***

2 **Il secondo passo da compiere per portare a termine la costruzione della classe '*libro*', consiste nel creare la classe e associargli un nome: dal menu *Inserisci* selezionare la voce *Classe*, digitare nel campo nome '*libro*' e premere *Applica.***

3 **Dal menu *Inserisci* selezionare *Attributo*, digitare nel campo nome *num_pagine* e premere *Applica.* Dal menu *Funzioni* selezionare la voce *Java | Genera Codice* e premere *Continua.* Selezionare la voce *Java | Compila Codice* e premere *Esci.* Selezionare la voce *Java | Esegui Applicazioni.***

Database Design Studio Lite 2.00.4e

Un completo sistema per la creazione e l'interrogazione di basi di dati.

Con un approccio completamente visuale, Database Design Studio consente anche agli utenti meno esperti di sfruttare le potenti caratteristiche del Enhanced Entity Relationship Diagram (EERD). Con il supporto per i più diffusi DBMS presenti sul mercato, DDS rappresenta una interessante soluzione "vendor independent" per la gestione di basi di dati. Tra i formati supportati segnaliamo ANSI, Ingres, InterBase, Informix, SQLBase, MicroSQL, Access, Microsoft SQL Server, MySQL e Oracle.

INIZIARE BENE

Per partire col piede giusto con Database Design Studio Lite è bene cominciare creando un nuovo progetto, dopo di che è possibile cominciare a lavorare subito sull'editor dei Diagrammi Entità-Relazione. È infatti da questo modello che discendono le altre proprietà controllabili dall'ambiente. Diciamo pure che tutte le altre viste fornite da DDS sono generate automaticamente, mentre il diagramma Entità-Relazione sarà il nostro banco operativo. Il cuore di DDS è il completo editor per la creazione e gestione dei diagrammi Entità-Relazione.
Il disegno dell'interfaccia e le opzioni disponibili sono frutto di un'attenta analisi delle richieste degli utenti della precedente versione. Altro importante elemento è l'editor per i Data Structure Diagram (DSD). Il DSD è automaticamente creato a partire dal diagramma Entità-Relazione e non è richiesto

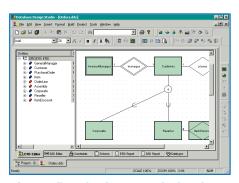



Fig. 1: L'interfaccia per manipolare lo schema Entità-Relazione.

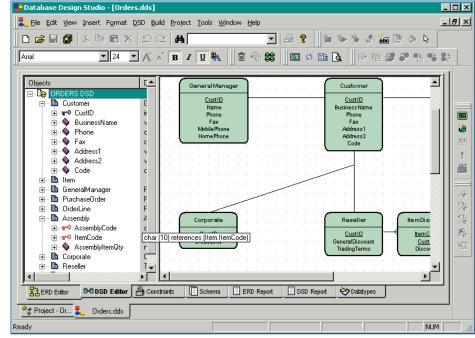



Fig. 2: Il Data Structure Diagram.

alcuno sforzo da parte dello sviluppatore. Rimarchevole risulta essere la gestione automatica delle chiavi esterne: l'utente non è dunque sollevato dal dover aggiornare manualmente le chiavi esterne corri delle varie entità del diagramma. Il modello Entità-Relazione non dovrà includere alcuna informazione sulle chiavi esterne, dovrà invece indicare esclusivamente le chiavi primarie, gli attributi comuni e le relazioni tra le varie entità. Database Design Studio, utilizzando le regole descritte nel diagramma Entità-Relazione, sarà in grado di distribuire le chiavi esterne nel Data Structure Diagram.

TOOL DEL PACCHETTO

All'interno di Database Design Studio troviamo alcuni interessanti tool per il test dei database: il Datastore editor consente di creare dei dati campione col solo scopo di mettere alla prova le funzionalità della base di dati. Il vantaggio consiste nell'avere una interfaccia "general pur pose" pronta per l'uso. Inoltre, una volta creato l'insieme di dati campione, non è necessario replicarlo per i vari DB che vogliamo testare: anche utilizzando DBMS differenti, tutte le operazioni di conversione sono a carico dell'ambiente. I filtri utilizzati per convertire i dati possono inoltre essere personalizzati per esigenze particolari. Una delle più interessanti funzioni implementate da Database Design Studio consiste nella produzione automatica di codice sorgente in Visual Basic, Java oltre allo scripting per tutti i più diffusi server di database. È possibile avviare e compilare il codice direttamente all'interno dell'ambiente. Da sottolineare che gli editor relativi ai vari linguaggi sono tutti dotati di syntax highlighting per una più facile lettura del codice.

SCHEDA TECNICA

Nome prodotto: Database Design Studio Lite 2.00.4e
Produttore: Chili Source
WEB: www.chillisource.com
Licenza: Trial
Prezzo della versione completa: $79.95
Sul CD: dds2004t.exe

In A Flash Pro 3.0

Un interessante tool per creare immagini e animazioni Flash pronte per le tue presentazioni.

In A Flash Pro 3.0 è stato creato per consentire agli sviluppatori di realizzare, in modo semplice e veloce, piccole presentazioni, CD multimediali, e filmati da visualizzare con colleghi, amici e parenti. Gli utenti possono scegliere gli elementi visuali da inserire nelle loro applicazioni, tra quelli presenti sul proprio PC. A queste immagini può essere aggiunto del testo, diverse animazioni, file audio e vari effetti: rotazione, dissolvenza... Bastano pochi passaggi per pubblicare le proprie applicazioni su NetGui oppure in locale.

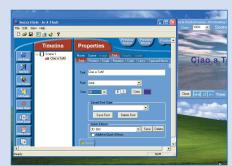
L'AMBIENTE

Appena si lancia l'applicazione, compaiono due finestre come in Fig. 1: quella principale e quella nella quale vengono visualizzate le anteprime. Analizziamo la finestra principale: oltre alla solita Barra dei Menu ed alla solita Barra degli Strumenti (apri un nuovo foglio di lavoro, un'applicazione esistente, salva…) distinguiamo due ambienti. Nell'area *Timeline*, sono visualizzati gli oggetti man mano che vengono creati e sono presenti ben otto pulsanti che ci consentono di creare una nuova applicazione, aggiungere del testo, effetti grafici, file audio, inserire/cancellare/copiare una scena, o eliminare un oggetto creato. Nell'area Property sono presenti diverse schede dalle quali è possibile scegliere colori, effetti speciali, dimensioni… insomma le proprietà degli oggetti che stiamo creando. Da qui è possibile aggiungere immagini, testo, animazioni, file audio e visualizzare tutti i componenti del progetto.

Fig. 1: **Scheramata principale dell'applicazione.**

SCHEDA TECNICA
Nome Prodotto: In A flash Pro 3.0
Produttore: NETGUI
WEB: www.netgui.com/product_flashpro.asp
Licenza: Trial
Prezzo: $99.95.
Sul CD: NgIaf30pTrial.exe

Realizziamo un'applicazione

1 **Cliccando nella casella di controllo *"Image"*, compare una finestra da cui è possibile scegliere l'immagine di sfondo per la nostra applicazione. Nella finestra *"Previewing Scene"* compare un'anteprima. Per aggiungere del testo, basta cliccare su *"Add Text"*.**

2 **Si attiverà la scheda *"Text"* dell'area *"Properties"*, qui possiamo scrivere il nostro testo, scegliere il colore, la dimensione ed eventuali effetti. Attivando la scheda *"Position"*, possiamo scegliere in quale posizione mettere il nostro testo.**

3 **Cliccando su *"Add Graphic"*, compare una finestra da cui è possibile scegliere un'animazione. Dalla scheda *"Special effect"* è possibile impostare effetti speciali. Per aggiungere effetti audio, basta cliccare su *"Add Audio"* e per la pubblicazione, su *"Publish"*.**

Fujitsu NetCOBOL for .NET Version 1.1

Il più potente tool di programmazione COBOL esistente.

Un compilatore COBOL creato appositamente per il Framework .Net di Microsoft: il risultato della compilazione è dunque in MSIL (Microsoft Intermediate Language) e può dunque essere eseguito dalla macchina virtuale, il Common Language Runtime. L'integrazione con .Net è completa e consente una stretta cooperazione con gli altri linguaggi previsti dalla piattaforma, senza contare l'interessantissima possibilità di utilizzare il COBOL come linguaggio di scripting per ASP.NET.

Fig. 1: Dopo l'installazione, all'interno di Visual Studio .NET avremo nuove voci fra cui scegliere.

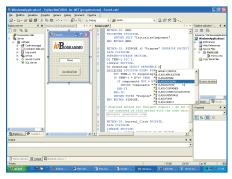
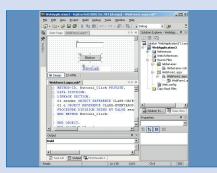
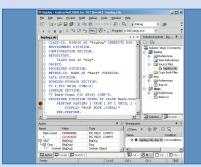
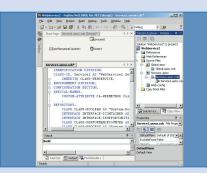



Fig. 2: La funzione intellisense dell'editor risulta aggiornata, e anche tutti gli altri elementi dell'IDE sono risultano arricchiti da nuove funzionalità.

Per utilizzare Fujitsu NetCOBOL, non è necessario avere installata una copia di Visual Studio .NET, è infatti incluso un completo IDE che fa da interfaccia con il compilatore. Se invece abbiamo installato Visual Studio, NetCOBOL si integrerà alla perfezione nell'ambiente, completandolo della possibilità di utilizzare un nuovo linguaggio per la creazione di un progetto. NetCOBOL consente dunque di utilizzare il COBOL nel mondo delle applicazioni Internet e dei Web Services. Da sottolineare la compatibilità verso COBOL-85, che viene compilato in modo da poter essere eseguito (così com'è) all'interno del Framework .NET: una grande occasione per per gli esperti del linguaggio che potranno valorizzare le loro conoscenze e accedere, senza sforzi, ai nuovi ambiti di programmazione.
La sintassi adottata dall'ambiente di Jujitsu è quella dell'Oject Oriented COBOL, in una versione che leggermente riadattata per andare in contro alle esigenze dell'ambiente .NET: le differenze sono davvero minime e si limitano per lo più all'aggiunta di nuove funzionalità.

SCHEDA TECNICA

Nome prodotto: NetCOBOL for .NET Ver. 1.1
Produttore: FUJITSU SOFTWARE CORPORATION
WEB: www.fsw.fujitsu.com/
Licenza: Trial valida trenta giorni
Prezzi: NetCOBOL for .NET Professional $2,340
NetCOBOL for .NET Developer $4,290
NetCOBOL for .NET Universal $5,460
Sul CD: NetCOBOLforNET_V1.1.zip

Un ambiente ricco di possibilità

1 Creare applicazioni Web è davvero semplicissimo: attraverso le stesse azioni di drag&drop utilizzate per costruire le interfacce di applicazioni Windows, è possibile realizzare complesse interfacce Web, tenendo sempre sotto controllo la complessità del progetto.

2 Per il debug, possiamo affidarci al Debugger integrato in VS.NET che consente di monitorare l'esecuzione del codice, evidenziando con una diversa colorazione la riga di codice corrente. Sono disponibili sofisticate funzioni condizionali per i breakpoint.

3 Ora anche i programmatori COBOL potranno dire la loro nel campo dei Web Services: NetCobol supporta il Web Services Designer che consente di assemblare per via visuale i componenti di un servizio, facendosi carico di generare il codice relativo in modo automatico.

ACE (Another C++ Editor) 2.0

Semplicità e linearità sono le caratteristiche fondamentali di questo editor gratuito orientato allo sviluppo di codice C++ e che si sposa alla perfezione con il compilatore a linea di comando della Borland. L'interfaccia si presenta scarna ma efficace e, grazie all'uso intensivo delle finestre tab, consente di gestire agevolmente progetti che coinvolgano numerosi file. Sono presenti numerose possibilità di personalizzazione e l'interfaccia non rinuncia a funzioni come il search&replace e la numerazione delle linee. Gratuito.

Nel CD: ace1.exe

ActiveInstall Professional 1.0

ActiveInstall è un potente strumento per lo sviluppo di pacchetti Windows Installer che, oltre a fornire il supporto per tutte le tecnologie attuali (compreso il .NET Framework), ha dalla sua la possibilità di essere programmato utilizzando Visual Basic for Applications (VBA). Una funzionalità davvero interessante e che permette di interagire agevolmente con l'applicativo utilizzando un linguaggio conosciuto da un vastissimo numero di persone. All'atto dell'installazione è necessario essere in possesso di uno user name e di una password, che si possono ottenere collegandosi al link *http://www.activeinstall.com/MyActiveInstall/Register.aspx*. Versione di prova valida dieci giorni.

Nel CD: AIProSF

BW*Wizard for Struts 5.33

Attraverso un massiccio utilizzo dei template forniti dal framework open source Jakarta Struts, BW*Wizard consente di costruire, compilare e testare applicazioni che facciano uso di transazioni in pochi istanti. Il codice prodotto dall'applicazione e pure Java ed è completamente ispezionabile e modificabile. BW*Wizard si presenta come un vero e proprio RAD (rapid application development) in cui la creazione delle applicazioni e completamente Model-driven: semplicemente indicando le tabelle ed i campi che si vogliono includere nell'interfaccia, si istruisce *BW*Wizard* sulle specifiche della applicazione che desideriamo creare. Il risultato consiste in applicazioni Struts-based che includono elementi JSP, ActionForm e struts-config.xml, tutti liberamente modificabili e adattabili ad esigenze più specifiche. Versione di prova valida quindici giorni.

Nel CD: BWStruts533

Data Conjure 1.1

Un utile tool che si occupa di generare dati di prova al fine di testare prestazioni e funzionalità di database. Attraverso una semplice ed intuitiva interfaccia grafica è possibile indicare il tipo di dato da generare ed una serie di regole sulla sua rappresentazione. Nella sua semplicità risulta un prodotto ben fatto ed è rimarchevole che, ad esempio, indicando come tipo di dato Nome, Cognome o numero di carta di credito, il programma non si limiti a generare delle stringhe casuali, ma fornisca dei nomi plausibili e dei codici compatibili con gli standard imposti dalle carte di credito. Versione di prova.

Nel CD: dataconjure.zip

DBScheduler 1.3

Una efficace utility che, attraverso un driver ODBC, riesce a pilotare le principali funzioni di un database, consentendo di automatizzare e schedulare una serie di attività che vanno dall'aggiornamento alla creazione di report. Utilissima la possibilità di salvare i risultati delle operazioni e inviarli via mail, sempre in modo automatico. I report generati possono essere ampiamente personalizzati e possono essere pubblicati in automatico come pagine HTML. Versione di valutazione valida quindici giorni.

Nel CD: dbscheduler.exe

Diff Doc Professional 2.24

Una piccola applicazione che consente di confrontare due porzioni di testo, evidenziandone le differenze. Supporta i numerosi formati (Word, Excel, RTF, PDF e Wordperfect) e si dimostra utile in svariate circostanze. Sono disponibili due modalità di visualizzazione: in un'unica finestra o con i due documenti affiancati. Versione di valutazione, presenta alcune limitazione nelle funzionalità.

Nel CD: MD.exe

i-Canvas 1.5

Un sistema di sviluppo per la creazione di siti e CD orientati all'e-learning, che non richiede alcuna conoscenza di programmazione. All'interno di un ambiente visuale completamente WYSIWYG (What You See is What You Get), è possibile impostare lezioni e sezioni, visualizzandole subito allo stesso modo in cui saranno presentate agli studenti. Grazie ad una serie di interfacce predefinite, è possibile essere subito produttivi mente, l'ampia possibilità di personalizzazione, garantisce un fine controllo sui risultati che si vogliono ottenere. La capacità di importare materiale sia da Word che da Power Point rappresenta una garanzia per chi ha già dei progetti avviati. Le capacità di registrazione e playback dell'audio sono presenti sia lato tutor che lato client e consentono un contatto "in differita" molto utile fra docente e allievi. Verisione di valutazione valida quindici giorni.

Nel CD: iCanvas_v1.5.zip

Java Web Services Developer Pack 1.1

Da Sun, la nuova release del fondamentale tool che, insieme alla piattaforma Java, consente agli sviluppatori di costruire, testare e pubblicare un vasto insieme di software diversi: applicazioni XML, Web Services e applicazioni Web. Il Java Web Services Developer Pack (o WSDP) include le implementazioni standard dei più diffusi protocolli per Web Services: WSDL, SOAP, ebXML e UDDI. Sono inoltre presenti delle implementazioni fondamentali per lo sviluppo di applicazioni Web come Java Server Pages, e la libreria standard di tag JSP. Questi standard Java consentono agli sviluppatori di inviare e ricevere messaggi SOAP, cercare e recuperare informazioni in registri UDDI ed ebXML, oltre a d permettere la rapida costruzione e la pubblicazione di applicazioni Web basate sui più aggiornati standard JSP.
Il Java Web Services Developer Pack v1.0_01 include:

- Java API for XML Messaging (JAXM) v1.1.1
- Java API for XML Processing (JAXP) v1.2.2
- Java API for XML Registries (JAXR) v1.0.3
- Java API for XML-based RPC (JAX-RPC) v1.0.3
- SOAP with Attachments API for Java (SAAJ) v1.1.1
- JavaServer Pages Standard Tag Library (JSTL) v1.0.3

- Java WSDP Registry Server v1.0_04
- Web Application Deployment Tool
- Ant Build Tool 1.5.1
- Apache Tomcat 4.1.2 container

Nel CD: jwsdp-1_1-win

NS Basic 3.0

Un completo ambiente di sviluppo per realizzare applicazioni Palm in un linguaggio semplice e accessibile come il BASIC. Le applicazioni possono essere testate su PC e quindi scaricare sul palmare. Facilissimo da utilizzare, sono disponibili numerosi oggetti pronti per l'uso e, grazie ai tanti progetti di esempio acclusi, è possibile imparare in poco tempo a realizzare applicazioni.

Nel CD: NsbasicDemo.exe

PDA Toolbox 5.2

Un ottimo tool che consente, anche ai meno esperti, di realizzare complesse applicazioni per palmari. Un'interfaccia completamente visuale ed un modello di programmazione completamente event-driven, consentono di ottenere sofisticate applicazioni con grande semplicità e con un potente supporto alle problematiche di accesso ai dati. Le applicazioni possono essere distribuite sia in versione Palm sia in versione Pocket PC. Versione di valutazione valida trenta giorni.

Nel CD: pdatv52a.exe

PowerVista Bridge Standard Edition 2002-1

Un ambiente per la creazione di DB-Application che, grazie ad un ampio e sapiente utilizzo di Wizard, assicura una incredibile rapidità di sviluppo. Il time-to-market risulta dunque ridottissimo e, grazie al supporto verso tutti i più diffusi DBMS (Oracle, SQL Server, Access, Informix, DB2, Interbase e altri), PowerVista potrà essere utilizzato con profitto da un ampia schiera di sviluppatori. Le applicazioni realizzate con PowerVista possono essere utilizzate indifferentemente da utenti singoli, in ambienti LAN o appoggiandosi a Internet. Versione di prova valida quattordici giorni.

Nel CD: pvbt2002

RMTrack Issue Tracking 1.0.2

Una applicazione Web-based che consente un'agevole gestione dello sviluppo di progetti software di grandi dimensioni e del relativo processo di bug tracking. Semplicità e flessibilità sono le linee guida dell'ambiente che, grazie ad un sistema grafico di workflow, consente di interagire per via visuale con grande immediatezza. Utile la possibilità di generare report direttamente in formato Excel. Versione di valutazione valida trenta giorni.

Nel CD: rmtrackv1.0.2.exe

Serial Monitor 2.25

Un'applicazione per il monitoraggio delle porte seriali che si rivela di grande utilità nelle operazioni di troubleshooting e che ha dalla sua la capacità di tenere in un traccia in un log di tutte le attività che hanno interessato una porta in un determinato arco temporale. Un'interfaccia particolarmente curata e l'ampio uso di wizard rendono l'utilizzo di questa applicazione particolarmente piacevole, oltre che efficace. Versione di valutazione valida trenta giorni e limitata ad un massimo di cento sessioni.

Nel CD: sermon.exe

Setup2Go 1.9.6

Per quanto spartano, questo ambiente offre tutte l'essenziale per realizzare completi pacchetti di installazione windows. Grazie ad un semplice e completo Wizard, anche chi non ha esperienza di programmazione può arrivare in poco tempo alla costruzione di pacchetti pronti per essere distribuiti. Gratuito.

Nel CD: setup2go.exe

SP Wizard for SQL 1.0

Grazie a questa applicazione è possibile risparmiare tempo prezioso nella generazione di Stored Procedure per Sql Server. Scelto il database, l'applicativo si collega analizzandone la struttura e proponendo, attraverso un semplice wizard, una serie di scelte. Al termine della procedura guidata sarà inoltre possibile generare codice per ADO.NET in C#, Visual Basic .Net e Managed C++.

Nel CD: SPWSetup.exe

W2XML 2.0

Un accessorio indispensabile nella cassetta degli attrezzi di uno sviluppatore: questo tool consente di convertire in modo automatico e veloce i file .doc in formato XML. Gli stili di Word sono riportati nel formato di destinazione e, grazie alla documenti XSLT personalizzabili, è possibile sfruttare a anche le informazioni associate allo stile. Tra le nuove funzionalità si segnala la generazione automatica di un file css per l'utilizzo immediato dei documenti XML in ambito Web. Per una corretta installazione, richiede che sia stato precedentemente installato il .NET Framework ed il pacchetto Universal Application Console, quest'ultimo riportato sul CD. Versione di valutazione della durata di quindici giorni, limitata alla conversione di 15 documenti.

Nel CD: Word2XML

Web Service Creator 1.2

Davvero comoda questa applicazione che si sostituisce al programmatore nella scrittura di noiose righe di codice, riuscendo a generare un completo Web Service a partire da un database. Una volta indicate le funzionalità che si vogliono implementare, Web Service Creator si occupa di generare tutto il codice sorgente in C#, pronto per essere pubblicato o, eventualmente, ulteriormente modificato per andare incontro ad esigenze più specifiche. Questa versione di valutazione si collega solo a database Access, mentre le versioni commerciali supportano anche SQL Server ed Oracle.

Nel CD: Web Services Creator (Access Edition).exe

X2Net WebCompiler 2.0

Se vi trovate nella necessità di pubblicare un e-book, un manuale o la demo di un sito, questo è il prodotto che fa per voi: WebCompiler riesce a convertire, in breve tempo e con un intervento minimo dal parte dello sviluppatore, le pagine HTML di un sito in una completa applicazione standalone. Con il supporto per JavaScript, Flash e cookies, offre una intelligente soluzione ad un ampio ventaglio di problematiche. La sicurezza è garantita dalla possibilità di protezione tramite password.

Nel CD: x2nwc.zip

XenoCode .NET

Un potente e flessibile sistema che consente di ottimizzare il codice prodotto per .NET, garantendo al contempo la protezione dai software di decompilazione che tentano di risalire al codice sorgente. Le operazioni di reverse engineering sono rese particolarmente ardue grazie ad un pesante mascheramento del controllo di flusso dell'applicazione, cosa che rende il risultato delle decompilazioni pressoché illeggibili a meno di non essere esperti dello spaghetti-code!

Nel CD: Setup.msi

Ed inoltre

Software sul CD di ioProgrammo

...di seguito sono riportati altri software contenuti all'interno del CD, raggruppati per categoria, con nome file e relativa estensione.

Ambienti di sviluppo

ACE (Another C++ Editor) 2.0
Un leggero editor per C++
ace1.exe

BLOBJ
Uno strumento C.A.S.E. di supporto alla produzione industriale di software
BlobJ

i-Canvas 1.5
Per creare siti e CD-ROM di e-learning
iCanvas_v1.5.zip

Microsoft DirectX 9.0 SDK
La nuova piattaforma per lo sviluppo di applicazioni grafiche
dx9sdk

NS Basic 3.0
Sviluppare applicazioni per Palm
NsbasicDemo.exe

Fujitsu NetCOBOL for .NET Version 1.1
Il più potente tool di programmazione COBOL esistente
NetCOBOLforNET_V1.1.zip

PDA Toolbox 5.2
Uno strumento visuale per realizzare applicazioni per palmari
pdatv52a.exe

Net Programming

Java Web Services Developer Pack 1.1
Il più completo ambiente per la programmazione di Web Services in Java
jwsdp-1_1-windows-i586.exe

In A Flash Pro 3.0
Un interessante tool per creare immagini e animazioni flash pronte per le tue presentazioni
NgIaf30pTrial.exe

Web Service Creator 1.2
Dal database al Web Service con un clic
Web Services Creator (Access Edition).exe

BW*Wizard for Struts 5.33
Per costruire una completa Web Application data-driven
BWStrutsWizard533.exe

Database

Data Conjure 1.1
Un sistema per testare database
dataconjure.zip

Database Design Studio Lite 2.00.4e
Un completo sistema per la creazione e l'interrogazione di basi di dati
dds2004t.exe

DBScheduler 1.3
Un sistema per schedulare le attività di gestione di un DB
dbscheduler.exe

Stored Procedure Wizard for SQL Server 1.0
Creare stored procedure in modo semplice e veloce
SPWSetup.exe

PowerVista Bridge Standard Edition 2002-1
Un mago per le Database Application
pvbt2002.exe

Installer

ActiveInstall Professional 1.0
Un completo ambiente per la costruzione pacchetti di installazione
AIProSF.exe

Setup2Go 1.9.6
Un piccolo ambiente gratuito per la creazione di Setup
setup2go.exe

Utility

W2XML 2.0
Per convertire documenti Word in XML
CNetWord2XMLTrial.msi

Code Co-op 4.0
Funziona in modalità distribuita senza la necessità di utilizzare un server!
co-op.exe

Diff Doc Professional 2.24
Controlla le differenze fra due testi
MD.exe

Serial Monitor 2.25
Controlla lo stato delle tue porte seriali
sermon.exe

RMTrack Issue Tracking 1.0.2
Tieni traccia dei bug delle tue applicazioni
rmtrackv1.0.2.exe

X2Net WebCompiler 2.0
Da HTML a EXE: ora e facile!
x2nwc.zip

XenoCode .NET 1.1
Proteggi il tuo codice .NET dalla decom pilazione
Setup.msi

Installazione ActiveX in Visual Basic

Dal menu Progetto selezionare la voce Componenti (CT RL+T); nella schermata presente a video è visibile una list box contenente l'elenco dei componenti ActiveX installati nel sistema; da questi è possibile selezionare uno o più componenti e confermare mediante il bottone OK; qualora il componente non fosse installato nel sistema ma fosse comunque presente nel computer è possibile selezionare quest'ultimo tramite l'utilizzo del bottone "Sfoglia" mediante il quale si ha accesso alle directory del sistema; da queste è possibile localizzare il componente da installare.

Risorse Java

Molte delle risorse Java riportate all'interno del CD ROM sono munite di file .java, .class e di file html per essere testate. Nel caso di compilazione del file .java si dovrà utilizzare un opportuno strumento, come ad esempio il JDK di Sun.
Per utilizzarlo si dovrà operare da prompt del DOS, accedere alla directory bin dell'ambiente stesso ed avviare il Java Compiler digitando la stringa: javac "nomefile".

LE FAQ DI IOPROGRAMMO

Le risposte alle domande più frequenti

Ogni mese troverete riportate le domande che più spesso giungono in redazione. Capita frequentemente che, affrontando linguaggi in continua evoluzione, si diano per scontati alcuni concetti o alcune caratteristiche di base: queste pagine sono l'occasione per ribadire o spiegare meglio tali nozioni.

Visual Basic

1 Che cosa sono le form di tipo modale, e che differenza c'e' tra una form modale ed una non modale?

Le form di tipo modale sono delle normali form che hanno in più la caratteristica di sovrapporsi a tutte le altre attive nell' applicazione in esecuzione, richiedono necessariamente un input dall' utente e limitano il modo di funzionare dell'applicazione finché non ricevono in input l'azione richiesta. In pratica una form modale ha sempre il focus attivo e permette di disattivarlo finché viene soddisfatta una richiesta, quindi viene chiusa e continua la normale esecuzione del programma. Un esempio tipico di form modale è la *MsgBox* oppure *InputBox*. Per capire meglio il funzionamento basta inserire la riga:

```
MsgBox("Hello world!")
```

e si vede subito il comportamento di una form modale. Per visualizzare una form modale basta usare la sintassi:

```
Form1.Show Modal
```

invece usando la normale proprietà *Show* senza aggiungere parametri la form verrà visualizzata in modalità *Modeless* cioè non modale. È da tenere presente che, nel caso in cui si stia sviluppando un' applicazione di tipo MDI, tutte le form *MDIChilds* non possono essere di tipo Modal, in questo caso per avere una form modale bisogne necessariamente inserire una form SDI all'interno del progetto.

2 Come fare per usare il tasto Invio per spostarsi tra gli oggetti di una form?

Chi sviluppava applicazioni ai tempi del Dos o anche chi semplicemente l'utilizzava ricorderà, magari con un po' di nostalgia, che il tasto *Invio* aveva la caratteristica funzione, oltre che confermare ed inviare l'input per essere elaborato, anche di far avanzare il focus al campo successivo. Con l'avvento di Windows e di sistemi più evoluti la stessa funzione viene affidata al tasto *Tab*, ma se volessimo tornare indietro ed usare *Invio*? Visual Basic permette di utilizzare per lo scopo l'evento sollevato in un oggetto, per esempio un *TextBox*, chiamato *KeyPress*.
Tale evento viene invocato ogni volta che si preme un tasto all'interno dell'oggetto, esso ci informa anche su quale tasto è stato premuto quindi possiamo individuare il tasto *Invio* che corrisponde al codice Ascii 13.
Il codice seguente illustra quanto detto:

```
Private Sub Text1_KeyPress(KeyAscii As Integer)
 If KeyAscii = 13 Then
   SendKeys "{TAB}"
   Beep
 End If
End Sub
```

3 Come creare una connessione per trasmettere dati sulla rete e far dialogare le nostre applicazioni?

Chi sviluppa applicazioni che devono essere eseguite in un ambiente di rete può trovare utile la possibilità di far comunicare i vari programmi in esecuzione sui pc connessi alla network, oltre che per le funzioni di condivisioni delle risorse di base già fornite dal sistema operativo, ma anche per scambiare informazioni di vario tipo o anche semplicemente per gestire un sistema di messaggistica come le Chat, molto note ed usate su internet. Insomma l'enorme utilità di tale tecnica trova limiti solo nella fantasia dello sviluppatore. Tutto questo è possibile grazie all'uso delle librerie *Winsock* fornite da Windows e messe a disposizione del programmatore anche da VisualBasic in modo molto semplice. Partiamo considerando che abbiamo già installato correttamente in protocollo di rete TCP/IP e che sappiamo cosa significano i termini host, ip e porta. Per utilizzare le librerie winsock dobbiamo inserire nel progetto il relativo componente chiamato Microsoft Winsock Control che è contenuto in *MSWINSCK.OCX*, per fare ciò basta scegliere la voce componenti nel menu *Progetto* e poi selezionare il componente. Creiamo una form ed inseriamoci dentro un controllo winsock e due textbox, una per il testo in arrivo ed una per il testo inviato, chiamate rispettivamente "entrata" ed "uscita" e tre *CommandButton*.
I *CommandButton* serviranno per assegnare al componente winsock la porta di ricezione dei dati, per assegnare al componente i dati dell'host al quale ci si vuole connettere e l'ultimo bottone per inviare il testo digitato nella textbox.
Andiamo ora ad inserire il codice negli eventi degli oggetti inseriti e ad illustrarne il funzionamento.
Innanzi tutto nell'evento load della form che contiene gli oggetti dimensioniamo una variabile che ci servirà per contenere in testo ricevuto:

```
Private Sub Form_Load()
Dim dati As String
End Sub
```

Nell' evento *click* del *CommandButton* chiamato *pchost* chiediamo in input la porta su cui il componente deve mettersi in ascolto ed attiviamo la ricezione

```
Private Sub pchost_Click()
porta = InputBox("digita la porta locale")
Winsock1.LocalPort = porta
Winsock1.Listen
End Sub
```

Nell' evento *click* del *CommandButton connetti* chiediamo in input all'utente il nome o l'indirizzo ip dell'host e la porta da utilizzare per la trasmissione, quindi assegniamo questi dati alle rispettive proprietà del componente winsock

```
Private Sub connetti_Click()
host = InputBox("Digita il nome dell'host o il suo indirizzo ip")
porta = InputBox("Digita la porta")
Winsock1.RemoteHost = host
Winsock1.RemotePort = Int(porta)
Winsock1.Connect
End Sub
```

Cliccando il *CommandButton* chiamato *Invia* tramite il metodo *SendData* di *winsock1* inviamo i dati all'Host collegato

```
Private Sub Invia_Click()
Winsock1.SendData (uscita.Text)
End Sub
```

Quando un utente chiede di connettersi con noi il componente winsock solleva un evento chiamato *ConnectionRequest*, qui possiamo scegliere se accettare o meno la connessione.
In questo caso accettiamo sempre tutte le richieste di connessione ma potremmo anche stabilire delle condizioni per eventualmente non accettarle.

```
Private Sub Winsock1_ConnectionRequest(ByVal requestID As Long)
If Winsock1.State <> sckClosed Then Winsock1.Close
Winsock1.Accept requestID
End Sub
```

L'evento *DataArrival* viene sollevato nel momento in cui il *winsock1* avverte che qualcuno ha inviato del testo al notro PC e quindi, tramite il metodo *GetData*, possiamo leggerlo e visualizzarlo in una textbox

```
Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
dati = ""
Winsock1.GetData dati, vbString, bytesTotal
entrata.Text = dati
End Sub
```

Al metodo *GetDAta*, come si può vedere dall'esempio, vengono passati tre parametri, il primo è la variabile che conterrà i dati, il secondo stabilisce il tipo di dati da ricevere e quindi come devono essere interpretati e l'ultimo parametro indica il numero di bytes da ricevere. Arrivati a questo punto abbiamo realizzato un semplice programma di comunicazione in tempo reale, naturalmente è solo un piccolo esempio ma può essere una buona base di partenza per implementare nuove caratteristiche nelle applicazioni sviluppate. E' da tener presente che il codice illustrato permette il collegamento di solo due utenti per volta, uno che rimane nella attesa di connessione e l'altro che richiede la connessione, per permettere più connessioni simultanee bisognerebbe creare un applicazione che facendo da server permetta di distribuire i messaggi, ma questo non è lo scopo di questo esempio.
Per fare delle prove non c'e' bisogno di avere due computer collegati i rete, basta eseguire due volte il programma compilato , scegliere una porta libera e impostare una copia come host digitando la porta scelta quando viene richiesto e nell'altra copia ciccare su connetti e alla richiesta dell'indirizzo del computer Host l'ip 127.0.0.1 e come porta la porta scelta prima.

4 Come visualizzare un avi in un' area prestabilita?

Capita spesso di vedere nei programmi delle aree di schermo che visualizzano una piccola animazione che può avere vari significati, dal semplice logo all'indicatore di attività che ci informa che il programma è in fase di elaborazione. Visual Basic permette di implementare questa funzione in modo molto semplice: innanzi tutto bisogna inserire nel progetto il componente chiamato *Microsoft Multimedia Control* che è contenuto in *MCI32.OCX* che gestisce i files multimediali, per fare ciò basta scegliere la voce componenti nel menu *Progetto* e poi selezionare il componente dalla lista. Adesso abbiamo tutti gli strumenti che servono per realizzare un esempio come descritto. Creiamo una form ed inseriamo un *CommandButton* e una *PictureBox* e la posizioniamo dove vogliamo che venga visualizzato il filmato. A questo punto nell'evento *click* del *CommandButton* inserito inseriamo le seguenti istruzioni:

```
With MMControl1
    .hWndDisplay = Picture1.hWnd
    .FileName = "c:\windows\clock.avi"
    .DeviceType = "AVIVideo"
    .Command = "open"
    .Command = "play"
End With
```

queste righe specificano l'area di visualizzazione, il nome del file da eseguire, il tipo di file, ed i comandi per aprire il file multimediale ed eseguirlo. A questo punto eseguendo il progetto e cliccando su *CommandButton* verrà eseguito il filmato, naturalmente il file multimediale può essere anche un file audio e quindi alla proprietà *DeviceType* di *MMcontrol1* deve essere assegnata la stringa "sequencer" per i file Midi o "waveaudio" per i file WAV.

5 Cosa è il passaggio di parametri nelle procedure, e come funziona?

Tutti i linguaggi di programmazione, e quindi anche VisualBasic,

permettono di utilizzare delle astrazioni funzionali dette procedure e funzioni, tali astrazioni ci permettono di semplificare lo sviluppo, potenziare la programmazione e la modifica del codice. Tutte le operazioni ripetitive o che comunque vengono utilizzare spesso possono essere scritte una sola volta e poi richiamate da un qualsiasi punto dell'applicazione, il processo di esecuzione farà un salto alla procedura richiesta completerà il lavoro e tornerà all'istruzione successiva a quella che l' ha invocata. L'uso di queste astrazioni rende anche più semplice la modifica e la correzione del codice in quanto l'aggiornamento di una procedura verrà automaticamente estesa a tutto il programma. La procedura chiamata può essere autonoma e quindi non comunicare col resto del programma ma svolgere un'azione e poi concludere il proprio lavoro, la maggior parte delle volte però avviene che il programma deve fornire alla procedura i dati su cui lavorare per produrre un risultato, tali dati vengono chiamati parametri. Consideriamo a questo punto una procedura di esempio:

```
Private Sub Esempio(parform As String)
 MsgBox(parform)
End Sub
```

Il passaggio di parametri può avvenire per passaggio di valore o per passaggio di riferimento. Il primo tipo, il passaggio per valore, è il metodo più diretto e consiste nel fornire alla procedura il valore vero e proprio , un esempio di passaggio per valore può essere

```
Esempio (“Hello world!”)
```

In questo modo la routine riceverà in input la stringa e quindi un valore costante cioè non modificabile. Il passaggio di parametri per riferimento invece permette al programmatore di non specificare il valore ma di fornire un riferimento e quindi una variabile o il suo indirizzo in memoria, così facendo la procedura riceverà il dato contenuto nella variabile e potrà utilizzarlo. Un esempio del passaggi per riferimento può essere il seguente

```
Dim frase as string
Frase=”Hellp world!”)
Esempio (frase)
```

Risulta chiaro che in questo modo la routine riceve un riferimento al valore e ne ricava successivamente il valore stesso.
Un vantaggio può essere dato, oltre dalla variabilità del dato in ingresso, dal fatto che la procedura può in questo modo anche elaborare un risultato e memorizzarlo nella locazione da dove ha prelevato il parametro semplicemente assegnando il nuovo valore alla variabile che identifica il parametro che si chiama parametro formale al contrario del parametro di dichiarazione che è chiamato parametro attuale.

6 Come posso sapere se le batterie del portatile sono cariche?

Quando si esegue un programma su di un portatile c'è sempre il rischio che le batterie esaurendo la carica facciano andare windows in standby o addirittura facciano spegnere di colpo il notebook, in questi casi lavorando con database o con altri tipi di archivi è possibile che si verifichi una perdita di dati oppure se si è collegati ad una rete può risultare interessante la possibilità di inviare un messaggio di avvertimento ad un altro computer connesso. Per ricavare l'informazione descritta dobbiamo inserire nel progetto il componente *SysInfo* contenuto in *SYSINFO.OCX*, per fare ciò basta scegliere la voce componenti nel menu Progetto e poi selezionare il componente. Per verificare se il notebook è connesso alla rete elettrica o alle batterie si può usare il seguente codice

```
Select Case SysInfo1.ACStatus
Case 0
    MsgBox "Non connesso alla linea elettrica"
Case 1
    MsgBox "Connesso alla linea elettrica"
End Select
```

È possibile ricavare la percentuale di carica delle batterie utilizzando il seguente codice:

```
Dim Carica As String
If SysInfo1.BatteryLifePercent <> 255 Then
    Carica = SysInfo1.BatteryLifePercent
    MsgBox Carica & "%"
Else
    MsgBox "Non rilevato"
End If
```

E se volessimo sapere quanto tempo ancora il notebook può lavorare prima di esaurire completamente la carica delle batterie? Lo ricaviamo col codice che segue

```
If SysInfo1.BatteryLifeTime <> &HFFFFFFFF Then
    MsgBox Format((TimeSerial(0, 0, SysInfo1.BatteryLifeTime)), "h:mm")
Else
    MsgBox "Non rilevato"
End If
```

7 Come rilevare se il programma è già in esecuzione?

In alcuni casi è importante non permettere più volte l'esecuzione di un programma, specie quando si usa una porta seriale o comunque una risorsa che non puo' essere condivisa pena il blocco del task e possibile perdita di dati. Per rilevare se il programma è già in esecuzione il VisualBasic ci viene in aiuto mettendoci a disposizione il comando

```
App.PrevInstance
```

Questa proprietà ritorna un valore positivo (TRUE) se il programma è già in esecuzione, altrimenti in caso contrario ritornerà un valore negativo (FALSE).
L'esempio che segue illustra come può essere utilizzato il comando descritto:

```
Private Sub Form_Load()
If App.PrevInstance Then
```

```
MsgBox "Il programma è già in esecuzione"
Unload Me
End If
```

Se si prova ad eseguire più volte l'applicazione che contiene il codice dell'esempio tutte le esecuzioni tranne la prima verranno terminate visualizzando il messaggio della *MessageBox*.

8 Come creare ed utilizzare un file di testo?

Sviluppando un'applicazione può essere utile salvare delle informazioni in modo da poterle richiamare in una successiva esecuzione, molte volte per l'esigua quantità di dati da memorizzare non è il caso di utilizzare database tramite ADO che appesantirebbe l'esecuzione non sarebbe una buona idea neanche utilizzare il registro di windows per evitare di inserire informazioni che magari possono venire modificate spesso e potrebbero così essere un rischio inutile. Possiamo risolvere il problema scegliendo di utilizzare un semplice file di testo per memorizzare tutto ciò che serve. Analizziamo le due fasi di utilizzo dei file di testo e cioè la scrittura e la lettura, iniziamo con la scrittura:

l'esempio che segue apre un file in modalità Output cioè in scrittura , se il file non esiste viene creato . Il file creato si chiamerà "Filetesto.txt" e verrà posizionato nella directory di default, dopodiché verrà scritto nel file il contenuto di TextBox1.Text ed infine verrà chiuso.

```
Open "MyFile.txt" For Output As #1
Print #1, TextBox1.Text
Close #1
```

La seconda fase consiste nell'operazione li lettura dei dati contenuti nel file creato, quindi viene aperto il file in modalità Input, successivamente viene letta una serie di stringhe che abbiamo quantificato in massimo dieci ed assegnate all' array di stringhe frase. E' da tenere presente che ogni singola riga per essere riconosciuta come tale deve terminare col carattere di ritorno a capo Carriage Return.

```
Dim temp as string
Dim frase(10) as string
open "MyFile.txt" For Input As #1
Do Until EOF(1)
conta=conta+1
        Line Input #1, frase(conta)
Loop
Close #1
```

9 Come posso utilizzare la porta seriale per comunicare col modem o altre periferiche seriali?

L'interfaccia seriale, presente in tutti i computer sia nuovi che datati, permette di comunicare con tutti i dispositivi che dispongono di tale interfaccia, siano essi altri computer, modem, stampanti e così via. L'uso più frequente, almeno fino all'avvento di interfacce di comunicazione più recenti e veloci, è il collegamento del computer con il modem e quindi in questi casi è molto utile poter gestire le interfacce seriali e dare la possibilità alla propria applicazione di utilizzare le periferiche ad esse collegate.
Il VisualBasic permette di accedere alle porte seriali tramite l'uso di un componente chiamato Microsoft Comm Control contenuto in MSCOMM32.OCX. Per sviluppare un piccolo esempio di come si realizzi una comunicazione seriale procediamo all' inserimento di tale componente nel progetto. A questo punto creiamo un form e posizioniamo un oggetto *MSComm*, due *TextBox* che serviranno rispettivamente a gestire la comunicazione, a visualizzare il testo in arrivo ed a contenere il testo da inviare.
L'esempio che segue abilita una connessione bidirezionale sulla porta seriale *comm1*:

```
Private Sub Form_Load()
  MSComm1.CommPort = 1
  MSComm1.Settings = "9600,n,8,1"
  MSComm1.RThreshold = 1
  MSComm1.PortOpen = True
End Sub
Private Sub MSComm1_OnComm()
Dim ricev As String
ricev = MSComm1.Input
If Len(ricev) Then  Text1.Text = Text1.Text & ricev
End Sub
Private Sub Text2_KeyPress(KeyAscii As Integer)
MSComm1.Output = Chr$(KeyAscii)
End Sub
```

In questo piccolo esempio viene usata l'interfaccia seriale comm1 con velocità di trasferimento impostata a 9600 bps, nessuna parità, 8 bit per byte più un bit di stop. Questi parametri devono essere uguali su ambedue le macchine, altrimenti non si riuscirà a trasmettere. Per fare un tentativo si possono collegare due computer tramite interfaccia seriale ma in questo caso bisogna tener presente che il cavo necessario è di tipo incrociato a differenza del cavo di collegamento dei modem ed altre periferiche che è di tipo dritto.

10 Quando usare Private per le dichiarazioni di variabili e di procedure?

Ogni volta che ciò è possibile, è bene usare *Private* per le dichiarazioni. Dichiarando *Private* le procedure e le variabili di un modulo, infatti, queste saranno accessibili soltanto nel modulo stesso:

```
Private variabile As Integer
Private Sub Procedura()
```

Conoscere a priori se una variabile o una procedura debba essere dichiarata *Private* o *Public* non è facile, specialmente se non si è valutato bene il problema al momento della redazione del codice. Se la variabile o la procedura è dichiarata *Public* quando non dovrebbe esserlo, si potrebbero creare problemi di eccessi di visibilità dell'oggetto in questione. Per fortuna, molti di questi problemi possono essere risolti con un'analisi automatica del codice.

11 Come si può ottimizzare la velocità di visualizzazione?

La velocità di visualizzazione di un programma è spesso più importante della sua reale velocità di esecuzione. Ecco alcuni suggerimenti per aumentare la velocità di visualizzazione di un'applicazione:

- Impostare la proprietà *ClipControls* sul valore *False*.
- Usare il controllo *Image* invece di *PictureBox* e *Label* invece di *TextBox* quando possibile.
- Nascondere i controlli quando si impostano le proprietà; ciò impedisce di effettuare ridisegni multipli.
- Mantenere nascosto un form precedentemente caricato per visualizzarlo velocemente. Questo metodo richiede un dispendio cospicuo di risorse di memoria.
- Usare priorità basse per i processi lunghi. Si possono realizzare processi a bassa priorità di esecuzione collocando opportunamente dei controlli *DoEvents*. Un ottimo posto dove collocare i *DoEvents* sono i cicli interni. Bisogna tenere presente, però, che l'utente potrebbe premere qualche tasto. Il processo a bassa priorità deve tener conto di ciò che l'utente può o non può fare durante la sua esecuzione. Impostare quindi le proprietà *Processing* col valore *True*.
- Usare barre di avanzamento di stato.
- Pre-caricare dati che serviranno successivamente. Per esempio, potete caricare i contenuti di una tabella di un database in un vettore.
- Usare la caratteristica *Show* degli eventi *Form_Load*.
- Semplificare lo startup form o usare uno splash screen.

C++

12 Quando vengono invocati i distruttori per le variabili locali?

Le variabili "non-static" sono distrutte automaticamente quando l'applicazione esce dal loro ambito di visibilità. Senza voler approfondire troppo il concetto di visibilità, possiamo affermare che l'ambito di vita di una variabile termina subito dopo il simbolo "}" in cui la variabile è stata dichiarata. Al momento dell'uscita da tale ambito, le variabili vengono distrutte dal compilatore chiamando il distruttore appropriato nella loro classe di appartenenza. Se l'oggetto ha allocato memoria (ad esempio un array), la distruzione rilascia la memoria precedentemente impegnata:

```
class vettore
{
private:
float *ptr;
public:
// costruttore
vettore(int n) { ptr = new float[n]; }
// distruttore
~vettore() { delete [] ptr; }
};
```

```
main()
{
// ...
{
vettore v(5); // alloca la memoria
// operazioni...
} // fine dell'ambito di visibilità dell'oggetto vettore
// l'oggetto vettore viene qui distrutto, e la memoria viene rilasciata
// ...
}
```

13 Come funzionano le espressioni separate da virgola?

Le espressioni separate da virgola derivano dal linguaggio C. Spesso queste istruzioni sono impiegate all'interno di clicli *for* e *while*. Le regole sintattiche di queste istruzioni, tuttavia, sono lungi da essere intuitive. In primo luogo, un'espressione può essere composta di una o più espressioni secondarie separate da virgola. Consideriamo, ad esempio, la seguente espressione:

```
if (++x, --y, cin.good())
```

L'istruzione condizionale *if* contiene tre espressioni separate da virgola. Il compilatore C++ si assicura che ciascuna delle espressioni venga valutata e che i relativi effetti avvengano. In questo caso, la variabile *x* viene incrementata e la variabile *y* è decrementata. Tuttavia, il valore dell'intera espressione separata da virgola è soltanto il risultato dell'espressione più a destra. Di conseguenza, l'istruzione if di sopra è vera solo se il metodo *cin.good()* ritorna *true*.
Consideriamo invece la seguente espressione:

```
int j=10;
int i=0;
while( ++i, --j)
{
//...
}
```

Nel ciclo *while*, la variabile *i* viene incrementata mentre la variabile *j* è decrementata. Il ciclo prosegue fintanto che il valore di *j* è maggiore di 0.

14 Come richiamare una funzione per dell'avvio di un programma?

Alcune applicazioni richiedono di invocare funzioni prima che la parte principale del programma venga eseguita. Ad esempio, la funzioni di *log* deve essere richiamata prima dell'esecuzione dell'applicazione. Il metodo migliore per invocare queste funzioni è di collocare la loro invocazione all'interno del costruttore di un oggetto globale dell'applicazione, questo perché gli oggetti globali sono costruiti prima che l'esecuzione passi al programma. Pertanto, le funzioni invocate in un costruttore di un oggetto globale verranno invocate prima della funzione *main()*.

Ad esempio:

```
class Login
{ public:
  Login()
  {  attiva_login(); }
};
Login log; // istanza globale
int main()
{ record * prec=Leggi_login();
  //.. codice dell'applicazione
}
```

L'oggetto globale log viene costruito prima che l'esecuzione passi alla funzione *main()*. Durante la costruzione, l'oggetto invoca la funzione *attiva_login()*. Quando l'esecuzione passa alla funzione *main()* dell'applicazione, è possibile quindi leggere i dati dal log file.

15 Cosa sono i puntatori ai membri?

Una classe può avere due categorie di membri: membri funzioni (metodi) e membri dati (variabili membro). Inoltre, esistono due tipi di puntatori: puntatori a metodi e puntatori a variabili membro. Gli ultimi sono i più diffusi poiché, generalmente, le classi non hanno variabili membro pubbliche.
Tuttavia, se utilizziamo codice C esistente che contiene dichiarazioni *struct* o classi che hanno variabili membro dati pubbliche, i puntatori a queste variabili possono rivelarsi molto utili. La sintassi per i puntatori a membri è una delle più complicate del linguaggio C++. L'uso di tali puntatori, però, rappresenta una caratteristica molto potente del linguaggio, perché permette di invocare una funzione membro di un oggetto senza conoscerne il nome.
Analogamente, è possibile usare un puntatore per esaminare o modificare il valore di una variabile membro senza conoscerne il nome.

Java

16 Come posso determinare il tipo di un carattere?

L'oggetto di sistema Character offre una serie di metodi statici per determinare se un determinato carattere è un numero, una lettera, uno spazio, etc. Ecco degli esempi:

```
Character.isDigit(myChar)       // true se myChar è una cifra numerica
Character.isLetter(myChar)      // true se myChar è una lettera alfabetica
Character.isLetterOrDigit(myChar) // true se myChar è una cifra o una
                                                              lettera
Character.isSpaceChar(myChar)  // true se myChar è un carattere di
                                                           spaziatura
Character.isLowerCase(myChar)          // true se myChar è minuscolo
Character.isUpperCase(myChar)          // true se myChar è maiuscolo
```

17 Come posso realizzare la creazione gestita di un oggetto?

Un metodo *factory* è un metodo statico che crea istanze di un oggetto e che vi permette così di valutare le condizioni per la creazione o meno dell'oggetto. Ecco un esempio:

```
class MyClass {
  public static create() {
    if ( boolCondiz ) { // verifico se va bene creare l'oggetto
      return new myClass();
    }
    return null;
  }
}
```

18 Quale comando usare per creare JAR "eseguibili"?

Indicando nel Manifest File di un JAR la main class da utilizzare, si può eseguire l'archivio direttamente con l'interprete *java.exe* che cercherà così il metodo statico *main(String[] args)* della classe indicata. Con Windows sarà sufficiente anche un doppio clic sul *.jar* per eseguirlo.
La dicitura da inserire nel manifest file è la seguente:

```
Main-Class: MyStartClass
```

19 È possibile trasformare un file Java in un EXE?

Un semplice tool chiamato *java2exe* vi consente di impacchettare il vostro codice Java in un eseguibile per Windows. È gratuito e lo potete scaricare da internet, facendo una ricerca su Google.It, ma non si tratta di un compilatore nativo, bensì di un semplice wrapper che comunque richiede una JVM sul sistema d'esecuzione.

20 Si possono decompilare le classi?

Il bytecode Java compilato è semplice da decompilare per riottenere il codice sorgente (ad eccezione di eventuali commenti, che vengono rimossi dalla procedura di compilazione). Questa operazione deve essere effettuata SOLO in accordo con le leggi di copyright e ci si può servire di tool gratuiti o commerciali disponibili su Internet ad esempio:

members.fortunecity.com/neshkov/dj.html

oppure

kpdus.tripod.com/jad).

21 Come posso determinare il sistema operativo?

Con questo semplice metodo potete ottenere il nome del sistema operativo su cui viene eseguito il vostro codice Java:

```
System.getProperties("os.name");
```

22 Qual è il metodo per inviare dei cookie ai browser?

Da una servlet o una JSP è possibile inviare un cookie tramite l'oggetto javax.servlet.http.Cookie. Il codice è il seguente:

```
Cookie ck = new Cookie(nome, valore); // nome e valore sono due stringhe
ck.setMaxAge(secondi); // durata del cookie in secondi
response.addCookie(ck); // accoda il cookie alla response
                                              HttpServletResponse
```

23 Come si possono formattare le date?

Per formattare una data utilizzate l'oggetto java.text.SimpleDateFormat, con cui si può creare una stringa pattern che specifica come debbano essere mostrati anno, mese, giorno, ora, etc.

```
SimpleDateFormat df = new SimpleDateFormat("d MMM yy");
String s = df.format(theDate); // formatta la data in theDate di tipo Date
```

24 È possibile impedire il caching delle pagine web?

Da una servlet o JSP potete richiedere che la pagina web non venga memorizzata nella cache del browser tramite i seguenti comandi:

```
response.setHeader("pragma","no-cache");
response.setHeader("Cache-Control","no-cache");
response.setDateHeader("Expires","0");
```

I primi due comandi fanno riferimento a diverse versioni HTTP, quindi è meglio includerli entrambi per maggiore compatibilità, mentre il terzo indica che la pagina scade subito e deve quindi essere letta dal server.

ColdFusion MX per J2EE Application Servers

25

di *Gianluca Pinelli*

Macromedia ColdFusion MX for J2EE consente agli sviluppatori di creare e distribuire applicazioni CFML (ColdFusion Markup Language) sui maggiori application servers conformi alle specifiche Java 2 Enterprise Edition. L'architettura di ColdFusion nella versione MX è stata completamente rinnovata: si basa su una infrastruttura 100% J2EE, supporta a pieno e nativamente standard quali XML, Xpath e Web services, ma soprattutto conserva le caratteristiche di semplicità e potenza del CFML. La linea di prodotti ColdFusion MX comprende una versione "standalone", caratterizzata da una infrastruttura J2EE "embedded" (basata su tecnologia Jrun), e una versione ideata per essere installata sui maggiori application servers J2EE.
Di quest'ultima versione ci occuperemo con una serie di rapide domande/risposte.

26 Che cos'è Macromedia ColdFusion MX per J2EE Application Servers?

Macromedia ColdFusion MX Server per J2EE Application Servers è a tutti gli effetti un'applicazione J2EE. Infatti viene installata sugli application servers J2EE come applicazione enterprise (.ear) fornendo servizi di sviluppo, distribuzione e gestione per applicazioni ColdFusion e/o Java.

27 Quali application servers J2EE sono supportati da Macromedia ColdFusion MX?

Macromedia ColdFusion MX per J2EE Application Servers supporta i maggiori application servers quali:

- Macromedia JRun 4
- IBM WebSphere Application Server Advanced Edition, version 4.0.3
- IBM WebSphere Application Server, version 5 (esclusa la versione Express).
- Sun ONE Web Server 6.02
- Sun ONE Application Server 7
- BEA WebLogic Server 6, 7

Inoltre è possibile installarlo su altri application servers conformi alle specifiche SUN J2EE 1.3.

28 Quali sono le caratteristiche peculiari di Macromedia ColdFusion MX?

Le caratteristiche peculiari che distinguono ColdFusion MX dagli altri ambienti di sviluppo scripting-based sono: la ormai leggendaria semplicità e potenza del CFML; un potente motore di ricerca full-text basato su tecnologia *Verità*; la capacità illimitata di integrazione grazie al supporto nativo di XML e WebServices; un potente motore per la generazione di grafici e report ed infine il supporto alla tecnologia Macromedia Flash Remoting.

29 Che tipo di database sono supportati?

Nella versione MX di ColdFusion, le connessioni ai database vengono effettuate tramite driver JDBC. ColdFusion MX installa driver proprietari per la connessione ai maggiori DB quali Microsoft SQL Server, Oracle, IBM DB2, Sybase, Informix, Access, mySQL. Inoltre è possibile collegare qualsiasi altro DB che abbia dei driver JDBC di tipo IV e su sistema operativo Windows sono supportati anche i driver ODBC (mediante un bridge JDBC/ ODBC).

30 Che grado di integrazione c'è con le API dello standard J2EE?

ColdFusion MX for J2EE si integra completamente con JSP e Servlet grazie alla condivisione delle variabili di sessione (inoltre è possibile utilizzare l'including e il forwarding tra pagine CFML, JSP e Servlet), si possono importare librerie JSTL per

estendere le funzionalità del markup language, richiamare JavaBean ed infine è possibile lavorare con gli EJB (o con altri componenti enterprise quali oggetti CORBA e COM).

31 Perché a uno sviluppatore ColdFusion dovrebbe interessare Macromedia ColdFusion MX per J2EE?

ColdFusion MX for J2EE permette di distribuire le applicazioni esistenti in CFML (o di realizzarne di nuove) su Application Server J2EE (ad esempio per far fronte a esigenze di integrazione legacy a livello enterprise o per spostare applicazioni mission-critical su infrastrutture J2EE). Scegliere una infrastruttura J2EE permette inoltre di usufruire delle sue numerose funzioni enterprise, nonché di estendere il CFML con la potenza di Java.

32 Perché a uno sviluppatore che lavora con Application Server J2EE dovrebbe interessare Macromedia ColdFusion MX per J2EE Application Servers?

ColdFusion MX per J2EE Application Servers rappresenta un layer aggiuntivo che va ad integrare (e/o a sostituire) le API messe a disposizione dallo standard J2EE (JSP/Servlet) per generare contenuti web dinamici. Utilizzare questo ulteriore layer significa beneficiare della rapidità di sviluppo e delle numerose features del CFML integrandolo con applicazioni J2EE esistenti o svilupparne di nuove con le funzionalità peculiari di ColdFusion MX (Flash Remoting, Charting&Graphics, Web services ecc.).

33 In che modo è possibile sfruttare l'architettura J2EE sottostante a Macromedia ColdFusion MX per J2EE Application Servers?

ColdFusion MX per J2EE Application Servers è a tutti gli effetti una applicazione Java enterprise (in particolare ColdFusion MX si installa nel container web). Ciò significa che ColdFusion MX beneficerà dei servizi messi a disposizione dall'ambiente di runtime quali: connessioni JDBC, servizio JNDI, servlet container, advanced thread management, load balancing, vertical ed orizontal scaling ecc.

34 Quali sono i vantaggi in termini di performance nella versione MX di Macromedia ColdFusion?

Nella versione MX di Macromedia ColdFusion le pagine CFML vengono compilate in byte code. Questo significa che alla prima richiesta di una pagina il Web Container si occupa di compilare la pagina CFML in Java bytecode. Le richieste successive di questa pagina verranno processate dal bytecode tenuto in cache con un incremento nelle performance delle applicazioni web. Ovviamente un update della pagina forzerà la ricompilazione della pagina.

35 Quali sono le differenze tra ColdFusion MX Server e ColdFusion MX per J2EE Application Servers?

ColdFusion MX per J2EE Application Servers differisce dalla versione "standalone" per l'infrastruttura di base rappresentata dall'application server J2EE scelto. In questo caso ColdFusion MX beneficerà delle performance, della robustezza e delle features peculiari di un application server di fascia enterprise (nei test svolti da Macromedia si è dimostrato come le performance di ColdFusion MX su IBM WebSphere 4.03 siano maggiori rispetto alla versione "standalone" in termini di tempi medi di accesso alle pagine).

36 Come migrare un'applicazione sviluppata con vecchie versioni di ColdFusion su Macromedia ColdFusion MX per J2EE Application Servers?

Applicazioni sviluppate sulla versione 5.0 di ColdFusion possono migrare sulle versioni MX senza difficoltà e nella maggior parte dei casi senza un update del codice. Alcuni tag presenti nella versione 5.0 nella versione MX sono diventati però deprecati o obsoleti (i tag per creare i grafici nella 5.0 sono stati completamente sostituiti con nuovi tag nella versione MX).

37 Per cominciare a sviluppare con Macromedia ColdFusion MX per J2EE Application Servers quale versione bisogna acquistare?

Non bisogna acquistare nessuna licenza per sviluppare con ColdFusion MX. In tutta la linea di prodotti Macromedia ColdFusion MX la versione trial una volta scaduta diventa "developer" e permette di sviluppare e testare applicazioni web sul localhost e su un ip remoto.

38 Quale ambiente di sviluppo è consigliabile usare per creare applicazioni in CFML?

Nella versione MX, ColdFusion si integra totalmente con Macromedia Dreamweaver MX creando un unico e potente ambiente di sviluppo RAD. Link per approfondimenti:

- **Macromedia CFMX 4 J2EE**
  http://www.macromedia.com/software/coldfusion/j2ee/
- **Macromedia CFMX 4 J2EE Download Trial version**
  http://www.macromedia.com/software/trial_download/
- **Documentazione OnLine su CFMX 4 J2EE**
  http://livedocs.macromedia.com/cfmxdocs/
- **ColdFusion on J2EE Dev Center**
  http://www.macromedia.com/desdev/mx/coldfusion/j2ee/
- **ColdFusion on J2EE WebForum**
  http://webforums.macromedia.com/coldfusion/

Programmare
UN VIDEOGIOCO PER IL CELLULARE

Nel corso di questo tutorial, prenderemo in esame lo sviluppo di un semplice videogioco destinato ai telefoni cellulari e a tutti quei dispositivi in grado di supportare la piattaforma J2ME di Sun Microsystems. Il tutorial si rivolge a chi già vanti un minimo di dimestichezza con la programmazione Java, nei suoi ambiti più classici. Insieme con le singole fasi di sviluppo del videogioco, saranno discussi gli strumenti di sviluppo necessari per programmare software destinato alla piattaforma J2ME.

Se programmare videogiochi è tra le vostre passioni, dovreste tenere d'occhio la recente piattaforma J2ME, che si sta ritagliando uno spazio significativo nell'ambiente del game programming. I videogiochi per cellulari, da semplice ninnolo quali erano stati concepiti, stanno divenendo sempre più dei veri e propri oggetti di culto. Inizialmente, i primi cellulari a disporre di questa interessante caratteristica, arrivavano con una manciata di giochi preinstallati, il cui parco non poteva essere esteso o modificato. Con i dispositivi dell'ultima generazione, la situazione è radicalmente mutata. Oggi, è possibile connettere ad Internet un piccolo device come un comune telefono cellulare, e quindi scaricare (gratuitamente o, come accade più spesso, dietro pagamento) nuove attrazioni ludiche da aggiungere a quelle preinstallate. Le possibilità hardware e software, a mano a mano che la tecnologia di consumo matura, si stanno facendo sempre più interessanti e degne di attenzione. Va inoltre considerato il fiorente business, in costante espansione, che muove il mondo del wireless game programming.

LA PIATTAFORMA J2ME

J2ME (*Java 2 Micro Edition*) è il nome della proposta di Sun Microsystems per la programmazione di applicazioni (non soltanto a carattere ludico) destinate ai cellulari, ai palmari e ad altre categorie di dispositivi con capacità di calcolo ridotte. Il linguaggio Java, come è lecito attendersi, sta trovando terreno fertile in un settore costituito da soluzioni hardware estremamente poco omogenee. Al giorno d'oggi, sono davvero molti i produttori ben disposti verso la piattaforma di Sun, soprattutto nel settore della telefonia mobile. Un marchio che dà enfasi all'affermazione è il celebre Nokia, che sta facendo di J2ME il principale strumento di sviluppo per i software dedicati ai propri prodotti di punta. In sostanza, J2ME è una versione ridotta della classica piattaforma Java 2 già disponibile per i desktop di ogni tipo (J2SE). Tuttavia, è bene non farsi ingannare dalle apparenze. Se, da un lato, gran parte dei pacchetti base di Java sono stati eliminati per dare minor peso alla Micro Edition, dall'altro sono state introdotte numerose nuove API, concepite appositamente per lo sviluppo mirato ai dispositivi wireless. Comunque sia, un programmatore già avvezzo alla distribuzione standard di Java farà davvero poca fatica nel riadattare le proprie conoscenze all'universo della Micro Edition.

PRIMA DI COMINCIARE

Per seguire e mettere in atto quanto sarà discusso in questo tutorial, è necessario disporre di tre differenti strumenti:

1. Un normale PC da casa o da ufficio, con sistema operativo **Windows** o **Linux**.
2. **J2SE SDK**, in altre parole la distribuzione standard di Java destinata ai programmatori. Benché la piattaforma mirata dal tutorial sia di tipo differente, è necessario disporre di questo pacchetto, giacché sarà utile il compilatore presente al suo interno. Questo articolo si rivolge a chi già vanta dimestichezza con la programmazione Java, quindi si suppone che il lettore non incontri problemi nel soddisfare tale requisito. In caso contrario, la pagina Web di riferimento è all'indirizzo: *http://java.sun.com/j2se/downloads.html*
3. **J2ME Wireless Toolkit.** Il Wireless Toolkit di Sun Microsystems contiene l'implementazione di riferimento di J2ME, insieme con un semplice

J2ME

File sul CD
\soft\codice\codici_snake.zip

J2ME nel mondo

Avete qualche dubbio sulla capillare diffusione raggiunta dalla piattaforma J2ME? Ecco una lettura che fa al caso vostro:

http://www.microjava.com/articles/perspective/zelos

J2ME

Programmare
un videogioco per il cellulare

ambiente di sviluppo che semplifica le operazioni di compilazione e verifica delle applicazioni wireless. All'interno del toolkit, inoltre, trovano posto diversi emulatori di telefoni cellulari, ed altri possono essere scaricati dai siti dei produttori dei più importanti device attualmente in commercio. Grazie ad essi, sarà possibile verificare il funzionamento delle applicazioni su una vasta gamma di dispositivi. Quanto mostrato in questo tutorial è stato sviluppato e verificato servendosi della versione 1.0.4_01 del toolkit. Potete scaricare il software consultando la pagina Web: *http://java.sun.com/products/j2mewtoolkit*

Una volta installato e configurato tutto l'occorrente, è possibile proseguire nella lettura del tutorial.

MYSNAKE

Le applicazioni J2ME destinate ai telefoni cellulari sono chiamate MIDlet, poiché riflettono un insieme di specifiche chiamato MIDP (*Mobile Information Device Profile*). Lo scopo di questo tutorial è guidare il lettore alla realizzazione di una prima MIDlet ludica, come premesso in apertura. Per meglio soddisfare lo scopo, si è scelto di mostrare quello che, per antonomasia, è il videogioco per cellulari: il celebre Snake. Realizzare l'ennesima versione di questo gioco, specie se per cause didattiche si mette in atto una riduzione delle caratteristiche di contorno, è di una semplicità disarmante. Ci avventureremo nei meandri di un progetto nominato *MySnake*. Prima di mettere mano al codice dell'applicazione, andandosi a confrontare con gli strumenti di J2ME, è bene allestire uno schema preliminare, che mostri come dovrà essere organizzata l'applicazione. Tale schema è riportato

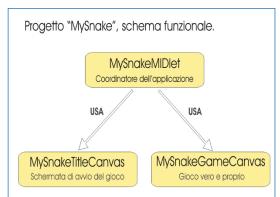



Fig. 1: Schema funzionale del prgetto MySnake.

in Fig. 1.

L'applicazione, come è possibile osservare, è stata scomposta in tre moduli indipendenti, che si tradurranno in altrettante classi Java. Il perno centrale è costituito da *MySnakeMIDlet*. Il compito di questa classe è coordinare le differenti fasi del gioco, facendo da tramite tra il dispositivo che esegue il gioco ed i restanti elementi del software. Sullo schermo saranno alternati i componenti *MySnakeTitleCanvas* e *MySnakeGameCanvas*. Il primo ingloba la schermata di avvio del gioco, che mostrerà il logo del software e disporrà di due comandi per l'interazione con l'utente: *"Esci"*, per uscire dal gioco, e *"Gioca"*, per avviare una nuova partita.
Quando l'utente richiederà l'inizio di una nuova partita, *MySnakeMIDlet* rimuoverà *MySnakeTitleCanvas* dallo schermo, inserendo al suo posto un'istanza di *MySnakeGameCanvas*. Questo componente, come è ora intuibile, incapsulerà la logica del gioco e realizzerà il motore grafico dello stesso. Ad esso, inoltre, saranno associati due comandi: *"Esci"*, per abbandonare l'applicazione come nel caso precedente, e *"Indietro"*, per tornare alla schermata del titolo.

UTILIZZO DEL WIRELESS TOOLKIT

Prima di passare all'implementazione di quanto proposto nel paragrafo precedente, è necessario guadagnare un minimo di confidenza con il Wireless Toolkit di Sun Microsystems. Il principale ambiente di sviluppo e test compreso al suo interno è nominato *KToolbar*. Avviate questo tool e

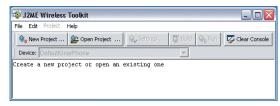



Fig. 2: Il J2Me Wireless Toolkit.

prendetene una prima visione (Fig. 2).
Per creare il progetto che andremo ad implementare, agite come segue:

1. Attivate il comando *"New Project"*.
2. Nominate il progetto *"MySnake"* e, in corrispondenza della voce *"MIDlet Class Name"*, inserite quello che sarà il nome della classe principale del progetto, vale a dire *"MySnakeMIDlet"*. Confermate l'operazione.
3. Nella finestra mostrata raggiungete la scheda *"MIDlets"*, evidenziate l'unica voce presente e preparatevi a variarne il contenuto con il tasto *"Edit"*.
4. Modificate la voce *"Icon"* da *"MySnake.png"* a *"/icons/MySnake.png"*. Questa voce serve per specificare l'icona associata all'applicazione. Ad ogni *MIDlet* è possibile abbinare un'icona differente, proprio come avviene con i programmi per i normali computer desktop. Co-

Che videogiochi si possono realizzare con J2ME?

Siete ansiosi di scoprire le potenzialità che J2ME riserva allo sviluppo di videogiochi? Provate ad esaminare alcuni dei titoli in commercio. Vi assicuro che rimarrete sorpresi!
Ad esempio, prendere in considerazione i titoli commercializzati da Gameloft, uno dei principali punti di riferimento nel settore:

http://www.gameloft.com/

me vedremo tra breve, collocheremo la nostra icona al percorso *"/icons/MySnake.png"*, per questo la modifica è necessaria.

5. Confermate ogni operazione effettuata.

A questo punto, il progetto *"MySnake"* è stato creato e configurato. In seguito, sarà possibile caricarlo sfruttando il tasto *"Open Project"* della KToolbar. Tutti i progetti sono ospitati all'interno della cartella che contiene l'installazione del Wireless Toolkit, nella speciale sottocartella *"\apps"*. Supponendo che la cartella che ospita l'installazione del toolkit sia *"C:\WTK104\"* (si ipotizza l'uso di un sistema Windows, ma sotto Linux è tutto analogo), dovreste ora poter localizzare la cartella che contiene il progetto *"MySnake"*, al percorso *"C:\WTK104\apps\MySnake\"*. Qui dentro distinguerete differenti cartelle, ognuna con il proprio scopo:

- **"\bin"**. Una volta completato, compilato ed impacchettato il gioco, il Wireless Toolkit metterà in questa cartella i file distribuibili.

- **"\classes"**. Qui vanno eventuali classi extra di cui si deve servire l'applicazione, sotto forma di semplici file .class di Java.

- **"\lib"**. Qui vanno eventuali classi extra di cui si deve servire l'applicazione, sotto forma di pacchetti JAR.

- **"\res"**. Qui vanno le risorse necessarie al funzionamento dell'applicazione. Questo è il posto ideale per posizionare immagini ed altri file di utilità.

- **"\src"**. Come il nome lascia intendere, il Wireless Toolkit desidera che i sorgenti della MIDlet vengano organizzati all'interno di questa cartella.

Per il nostro progetto ci serviremo solamente delle cartelle *"\res"* e *"\src"*. Nella prima bisogna posizionare le risorse di cui necessita l'applicazione, tipicamente immagini e file di generica utilità, mentre nella seconda vanno conservati i sorgenti del software. Create le cartelle *"\res\icons"* e *"\res\images"*, e sistemate al loro interno i file PNG che trovate nel pacchetto ZIP allegato all'articolo (nel CD-Rom che accompagna la rivista). Questi file sono indispensabili per un completo funzionamento della MIDlet. Dentro *"\res\icons"* va il file *"MySnake .png"*, che come anticipato rappresenta l'icona che sarà associata alla MIDlet. In *"\res\images"*, invece, vanno le tre immagini *"mysnake_bw .png"*, *"mysnake_graytones.png"* e *"mysnake_colors.png"*. Questi sono i loghi di cui ci serviremo all'interno di *MySnakeTitleCanvas*, come avremo modo di approfondire in seguito.

Prepariamoci ora a riempire la cartella *"/src"*, sviluppando i sorgenti del gioco.

MYSNAKEMIDLET

Andiamo a sviluppate il primo modulo dell'applicazione, costituito dalla classe *MySnakeMIDlet:*

```
// File: MySnakeMIDlet.java
// Pacchetti di J2ME necessari alla classe.
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
// MySnakeMIDlet estende MIDlet e implementa
                                    CommandListener.
public class MySnakeMIDlet extends MIDlet
                         implements CommandListener {
 // Queste costanti identificano i vari dispositivi.
 public static final int COLORS_DEVICE = 0;
                                  // Schermo a colori.
 public static final int GRAYTONES_DEVICE = 1;
                            // Schermo a toni di grigio.
 public static final int BW_DEVICE = 2;
                           // Schermo in bianco e nero.
 // Qui sarà memorizzato il tipo di dispositivo in uso,
                                    in corrispondenza
 // delle tre differenti costanti sopra dichiarate.
 private int deviceType;
 // La canvas che conterrà e mostrerà la schermata
                                    di avvio del gioco.
 private MySnakeTitleCanvas titleCanvas = null;
 // Questa canvas conterrà il gioco vero e proprio.
 private MySnakeGameCanvas gameCanvas = null;
 // I comandi ammessi dalla MIDlet per l'interazione
                                         con l'utente.
 private Command cmdExit  = new Command("Esci",
                                    Command.EXIT, 1);
 private Command cmdStart = new Command("Gioca",
                                  Command.SCREEN, 2);
 private Command cmdBack  = new Command(
                          "Indietro", Command.BACK, 2);
 // Costruttore. Prepara gli elementi di base per l'avvio
                                            del gioco.
 public MySnakeMIDlet() {
  // Acquisisce informazioni sul dispositivo in uso.
  Display display = Display.getDisplay(this);
  if (display.isColor()) {
   deviceType = COLORS_DEVICE;
  } else if (display.numColors() >= 256) {
   deviceType = GRAYTONES_DEVICE;
  } else {
   deviceType = BW_DEVICE;
  }
  // Crea la canvas da visualizzare all'avvio.
  titleCanvas = new MySnakeTitleCanvas(deviceType);
  // Aggiunge i comandi "Esci" e "Gioca".
  titleCanvas.addCommand(cmdExit);
  titleCanvas.addCommand(cmdStart);
  // Imposta il gestore di eventi per i comandi.
```

J2ME

Programmare
un videogioco per il cellulare

Nokia e J2ME

Nokia è tra i principali sostenitori di J2ME. Sul sito di Nokia dedicato agli sviluppatori, potrete reperire documentazioni e tool di sviluppo specializzati:

http://www.forum.nokia.com

J2ME

Programmare
un videogioco per il cellulare

```
  // Vedere il metodo commandAction() di questa
                                        stessa classe.
  titleCanvas.setCommandListener(this);
 }
 // Questo metodo gestisce i comandi azionati dall'utente.
 public void commandAction(Command c, Displayable d) {
   if (c == cmdExit) {
     // Chiusura della MIDlet.
     shutdown();
   } else if (c == cmdStart) {
     // Avvio di una nuova partita.
     if (gameCanvas == null) {
       gameCanvas = new MySnakeGameCanvas(
                                          deviceType);
       gameCanvas.addCommand(cmdExit);
       gameCanvas.addCommand(cmdBack);
       gameCanvas.setCommandListener(this);
     }
     Display display = Display.getDisplay(this);
     display.setCurrent(gameCanvas);
     gameCanvas.startGame();
   } else if (c == cmdBack) {
     // Torna alla schermata del titolo, se possibile.
     if (gameCanvas.stopGame()) {
       Display display = Display.getDisplay(this);
       display.setCurrent(titleCanvas);
     }
   }
 }
 // Questo metodo viene eseguito non appena la
                                       MIDlet è avviata.
 public void startApp() {
   Display display = Display.getDisplay(this);
   display.setCurrent(titleCanvas);
 }
 // Metodi la cui implementazione è richiesta dalla
                                         classe MIDlet.
 // In questo caso sono vuoti perché non si intende
                                      usufruire delle loro
 // funzionalità.
 public void pauseApp() {}
 public void destroyApp(boolean b) {}
 // Termina l'esecuzione della MIDlet.
 public void shutdown() {
   destroyApp(false);
   notifyDestroyed();
 }
}
```

Il package *javax.microedition.midlet* contiene la sola classe *MIDlet*. Qualsiasi MIDlet, affinché possa essere ritenuta tale e quindi eseguita in un ambito MIDP, deve estendere questa classe. *MIDlet* è una classe astratta, che richiede sempre l'implementazione di tre metodi:

- **public void startApp()**. Viene richiamato automaticamente non appena la MIDlet è stata creata. Deve contenere gli elementi necessari per l'avvio dell'applicazione.

- **public void pauseApp()**. Viene richiamato automaticamente quando la MIDlet deve essere messa in pausa, ad esempio perché si riceve una telefonata mentre si sta facendo uso dell'applicazione. Al suo interno deve essere posizionato il codice necessario per gestire l'eventuale interruzione improvvisa del software, facendo in modo che lo stato attuale possa essere successivamente ripristinato. Quando l'utente torna al programma dopo l'interruzione, viene nuovamente ed automaticamente chiamato startApp(). Per giovare alla semplicità di questo esempio introduttivo, si è stabilito di non dotare MySnake di tale caratteristica. Un gioco destinato al mercato, ad ogni modo, dovrebbe farne uso.

- **public void destroyApp(boolean b)**. Questo metodo viene richiamato immediatamente prima della chiusura definitiva della MIDlet. L'argomento booleano accettato da destroyApp() indica la necessità dell'operazione. Se è true, l'applicazione deve essere chiusa immediatamente, senza possibilità di ripensamenti. In caso contrario, il corpo del metodo può lanciare una MIDletStateChangeException, per richiedere di posticipare la chiusura dell'applicazione. Questa funzionalità è utile nel caso in cui la chiusura venga comandata in un momento critico, ad esempio durante la scrittura di dati. Per come è stato strutturato il nostro progetto, non ci saranno elementi da terminare prima della chiusura del gioco, né esisteranno casi in cui si preferirà andare contro quanto comandato. Pertanto, il corpo del metodo è vuoto.

I metodi *pauseApp()* e *destroyApp()*, in MySnakeMIDlet, hanno implementazione vuota, per i motivi già esposti. Al contrario, *startApp()* si cura dell'avvio del gioco in maniera coerente e completa.
Procediamo con ordine, osservando il ciclo di vita tipico di una MIDlet. Per prima cosa, è richiamato il costruttore della classe.
Nel nostro caso:

```
public MySnakeMIDlet() {
 // Acquisisce informazioni sul dispositivo in uso.
 Display display = Display.getDisplay(this);
 if (display.isColor()) {
   deviceType = COLORS_DEVICE;
 } else if (display.numColors() >= 256) {
   deviceType = GRAYTONES_DEVICE;
 } else {
   deviceType = BW_DEVICE;
 }
 // Crea la canvas da visualizzare all'avvio.
 titleCanvas = new MySnakeTitleCanvas(deviceType);
```

Emulatori aggiuntivi

Potete arricchire il parco di emulatori compresi nel vostro Wireless Toolkit, semplicemente scaricandone ed installandone di nuovi. Un esempio è quello fornito da Sony Ericsson partendo dall'indirizzo:

http://www.ericsson.com/mobilityworld/sub/open/technologies/java/index.html?PU=java

```
// Aggiunge i comandi "Esci" e "Gioca".
titleCanvas.addCommand(cmdExit);
titleCanvas.addCommand(cmdStart);
// Imposta il gestore di eventi per i comandi.
// Vedere il metodo commandAction di questa stessa
                                          classe.
titleCanvas.setCommandListener(this);
}
```

Qui vengono acquisite informazioni sul dispositivo in uso, attraverso la classe *Display* del package *javax.microedition.lcdui* (per certi versi, questo package è la controparte J2ME di *java.awt*).
Nella proprietà privata *deviceType* viene memorizzato un valore costante che può essere:

- **COLORS_DEVICE**, se il dispositivo fa uso di uno schermo a colori.
- **GRAYTONES_DEVICE**, se il dispositivo fa uso di uno schermo con almeno 256 toni di grigio.
- **BW_DEVICE**, nel caso di uno schermo in bianco e nero.

Il contenuto di *deviceType* potrà essere successivamente consultato, in modo che il software possa intraprendere strade diverse a seconda dell'hardware riscontrato. Sempre all'interno del costruttore, è inizializzato e configurato un oggetto *MySnakeTitleCanvas*, che analizzeremo a breve. A questo vengono aggiunti due comandi di interazione con l'utente: *"Esci"* e *"Gioca"*. Una volta disposti i comandi, è necessario impostare un gestore di eventi ad essi collegato, in altre parole una classe che implementi l'interfaccia *CommandListener*, dando corpo al metodo *commandAction()* da questa richiesto. *MySnakeMIDlet*, oltre ad estendere la classe *MIDlet*, implementa l'interfaccia *CommandListener* e definisce il metodo *commandAction()*. Quindi, è del tutto logico scrivere:

```
titleCanvas.setCommandListener(this);
```

Quando l'utente attiverà uno dei comandi abbinati all'istanza di *MySnakeTitleCanvas*, verrà automaticamente richiamato il metodo *commandAction()* di *MySnakeMIDlet*. Terminata l'esecuzione del costruttore, è il turno del già discusso *startApp()*:

```
public void startApp() {
  Display display = Display.getDisplay(this);
  display.setCurrent(titleCanvas);
}
```

L'operazione effettuata, in questo caso, è molto semplice: sullo schermo viene mostrata l'istanza di *MySnakeTitleCanvas* generata in precedenza. Sono visualizzabili tutte quelle classi che estendono, direttamente o indirettamente, *javax.microedition.lcdui.Displayable.* Tanto *MySnakeTitleCanvas* quanto *MySnakeGameCanvas*, come vedremo, soddisfano tale requisito, pertanto possono produrre e mostrare un output grafico. Passiamo ora in rassegna il metodo *commandAction()*, il cui compito è gestire le richieste che l'utente esprime attraverso l'uso di oggetti *Command:*

```
public void commandAction(Command c, Displayable d)
{
 if (c == cmdExit) {
  // Chiusura della MIDlet.
  shutdown();
 } else if (c == cmdStart) {
  // Avvio di una nuova partita.
  if (gameCanvas == null) {
   gameCanvas = new MySnakeGameCanvas(
                                    deviceType);
   gameCanvas.addCommand(cmdExit);
   gameCanvas.addCommand(cmdBack);
   gameCanvas.setCommandListener(this);
  }
  Display display = Display.getDisplay(this);
  display.setCurrent(gameCanvas);
  gameCanvas.startGame();
 } else if (c == cmdBack) {
  // Torna alla schermata del titolo, se possibile.
  if (gameCanvas.stopGame()) {
   Display display = Display.getDisplay(this);
   display.setCurrent(titleCanvas);
  }
 }
}
```

Se ad essere attivato è il comando *"Esci"* (*cmdExit*), il controllo del flusso di esecuzione sarà passato al metodo *shutdown()*:

```
public void shutdown() {
 destroyApp(false);
 notifyDestroyed();
}
```

La sequenza introdotta al suo interno è la maniera tipica per richiedere il termine e la chiusura della MIDlet corrente. La chiamata a *destroyApp()* è ridondante, giacché sappiamo che il metodo ha corpo vuoto. Tuttavia, in vista di estensioni future, è bene rispettare la regola. Il metodo *notifyDestroyed()*, definito all'interno di *MIDlet*, termina effettivamente l'esecuzione della *MIDlet*. Più complessi sono i codici associati ai comandi *"Gioca"* (*cmdStart*) e *"Indietro"* (*cmdBack*). Il primo prepara un'istanza di *MySnakeGameCanvas*, abbinandogli i due comandi *"Indietro"* e *"Esci"*. L'istanza del gioco viene mostrata sullo schermo, andando così a sostituire la schermata con il titolo. Il gioco viene

Documentazione ufficiale

La documentazione ufficiale del Wireless Toolkit e delle API di J2ME è compresa nella cartella "\docs", nella directory che ospita lo stesso toolkit.

J2ME

Programmare
un videogioco per il cellulare

avviato dal metodo *startGame()*, che dovremo implementare in seguito all'interno di *MySnakeGameCanvas*. Nel caso l'utente sia in fase di gioco, sullo schermo sarà presente il comando *"Indietro"*. Il gestore di eventi associato all'attivazione del comando, come è possibile osservare, tenta l'interruzione del gioco con *stopGame()*. Tale metodo dovrà restituire true nel caso il blocco sia possibile. Nel caso l'operazione possa essere eseguita senza problemi, la MIDlet torna a visualizzare la schermata con il titolo del gioco.
Questa preliminare analisi di *MySnakeMIDlet* ci offre importanti informazioni su come dovranno essere organizzati gli altri due moduli dell'applicazione:

- *MySnakeTitleCanvas* dovrà essere una classe visualizzabile, che estende *Displayable*.

- *MySnakeGameCanvas* dovrà essere una classe visualizzabile, che estende *Displayable*, ed inoltre dovrà implementare due metodi: public void *startGame()* e *public boolean stopGame()*, rispettivamente per l'avvio e l'interruzione di una partita.

Per un'analisi più approfondita dei contenuti dei package *javax.microedition.midlet* e *javax.microedition.lcdui*, fate riferimento alla documentazione ufficiale delle API di J2ME, distribuita insieme allo stesso Wireless Toolkit (vedi box laterale titolato *"Documentazione ufficiale"*).

PROBLEMATICHE DI SVILUPPO

Apriamo un breve interludio, dedicato alle principali problematiche che vanno affrontate quando si programma con J2ME.
Sostanzialmente, bisogna fare i conti con due elementi:

1. La possibilità di determinare solo a tempo di esecuzione le caratteristiche peculiari del dispositivo in uso.

2. La disponibilità di scarse risorse software e hardware.

Il primo punto elencato rappresenta il rovescio della medaglia di tutta la programmazione multipiattaforma, che trova particolare enfasi nel caso di J2ME e dei dispositivi wireless. Una MIDlet può essere eseguita da qualsiasi dispositivo supporti J2ME. Ovviamente, non tutti i device che supportano le specifiche di Sun hanno equivalenti requisiti hardware. In generale, bisogna fare i conti con alcuni parametri, tra i quali spiccano le caratteristiche dello schermo montato sul dispositivo.
Alcuni cellulari, ad esempio, usano schermi a colori, altri a toni di grigio ed altri ancora sono fermi al bianco e nero. Anche le dimensioni dello schermo vanno tenute in considerazione, poiché non sono le stesse per ogni dispositivo. Un palmare, ad esempio, dispone di un'area molto vasta, mentre un cellulare avrà sicuramente un video di dimensioni ridotte. Questi parametri vanno sempre tenuti in considerazione, soprattutto nel momento in cui l'oggetto di studio è un videogioco. Se si intende realizzare qualcosa di realmente multipiattaforma, è necessario disegnare un software adattabile, che risponda in maniera ottimale ad ogni possibile situazione. Per questo motivo, all'interno di *MySnakeMIDlet*, sono state acquisite e conservate le informazioni relative allo schermo in uso, che possono essere desunte dalla classe *Display*. Sempre per la stessa ragione, nella cartella "/res/images" abbiamo posizionato tre differenti loghi del gioco, uno destinato agli schermi a colori, uno a quelli a toni di grigio ed il rimanente ai display in bianco e nero. Ora che andremo a considerare le due classi del gioco incaricate di disegnare sullo schermo, non potremo prescindere da queste importanti considerazioni.

MYSNAKETITLECANVAS

MySnakeTitleCanvas è una classe molto semplice, anche se ci porta a confrontarci per la prima volta con le tematiche illustrate nel paragrafo precedente. Tutto quello che bisogna fare, è mostrare un logo sullo schermo. Il logo deve essere differente in base alle caratteristiche del display in uso:

```
// File: MySnakeTitleCanvas.java
// Pacchetti utili alla classe.
import javax.microedition.lcdui.*;
import java.io.*;
// MySnakeTitleCanvas estende la classe Canvas.
public class MySnakeTitleCanvas extends Canvas {
 // Buffer grafico ad uso interno.
 private Image buffer;
 private Graphics b;
 // Costruttore.
 public MySnakeTitleCanvas(int deviceType) {
  // Creazione del buffer e del relativo contesto grafico.
  buffer = Image.createImage(getWidth(), getHeight());
  b = buffer.getGraphics();
  // L'immagine buffer viene riempita di bianco.
  b.setGrayScale(255);
  b.fillRect(0, 0, getWidth(), getHeight());
  // Sceglie il logo in base al dispositivo.
  String imgPath = null;
  switch (deviceType) {
  case MySnakeMIDlet.COLORS_DEVICE:
   imgPath = "/images/mysnake_colors.png";
```

Sul Web

In rete, è possibile reperire numerose informazioni, tanto sulla programmazione di videogiochi quanto sulla piattaforma J2ME. Ecco alcuni dei principali punti di riferimento:

http://www.microjava.com/

http://www.jguru.com/faq/J2ME/

http://www.onjava.com/onjava/wireless/

http://www.gamasutra.com/

J2ME

Programmare *un videogioco per il cellulare*

```
        break;
    case MySnakeMIDlet.GRAYTONES_DEVICE:
        imgPath = "/images/mysnake_graytones.png";
        break;
    case MySnakeMIDlet.BW_DEVICE:
        imgPath = "/images/mysnake_bw.png";
        break;
    }
    // Carica l'immagine.
    try {
        Image img = Image.createImage(imgPath);
        // Se l'immagine è troppo ampia per lo schermo in
                                          uso si lancia
        // un'eccezione.
        if (getWidth() < img.getWidth() || getHeight()
                                  < img.getHeight()) {
          throw new Exception();
        }
        // Inserisce l'immagine al centro del buffer.
        b.drawImage(
                    img, getWidth() / 2,
                    getHeight() / 2,
                    Graphics.HCENTER | Graphics.VCENTER
                    );
    } catch (Exception e) {
        // In caso di eccezione (impossibile caricare
                                  l'immagine o schermo
        // troppo piccolo) il logo viene sostituito da una
                                      semplice scritta.
        b.setGrayScale(0);
        b.drawString(
          "MySNAKE",
          getWidth() / 2,
          getHeight() / 2,
          Graphics.BASELINE | Graphics.HCENTER
        );
    }
  }
  // Questo metodo, richiesto da Canvas, stampa il
                                       buffer sul video.
  public void paint(Graphics g) {
    g.drawImage(buffer, 0, 0, Graphics.LEFT |
                                        Graphics.TOP);
  }
}
```

MySnakeTitleCanvas estende la classe *Canvas*, che a sua volta estende *Displayable*. Quindi, indirettamente, *MySnakeTitleCanvas* estende *Displayable*, come richiesto. La classe *Canvas* contiene l'infrastruttura ideale per il disegno libero sullo schermo, l'ideale nel caso di sviluppo di un videogioco. Proprio come già avviene con l'AWT di J2SE, la classe *Canvas* richiede l'implementazione del metodo *paint()*, per l'organizzazione dell'output grafico. In questo caso, quindi, J2ME non si discosta molto da suo fratello maggiore. Il costruttore della classe acquisisce le informazioni comprese nell'argomento *deviceType* fornito dalla MIDlet, e quindi carica l'immagine più idonea al caso. Inoltre, la classe acquisisce per proprio conto le informazioni relative alle dimensioni dello schermo, attraverso i metodi *getWidth()* e *getHeight()* ereditati da *Canvas*.

CONCLUSIONI

Giunti a questo punto, sussistono quattro casi:

1. Deve essere mostrato il logo a colori.
2. Deve essere mostrato il logo a toni di grigio.
3. Deve essere mostrato il logo in bianco e nero.
4. Nessuno dei loghi escogitati può essere mostrato.

Il quarto caso può ricorrere in due distinte situazioni:

1. Quando l'immagine, per qualche motivo, non può essere caricata (lancio di una *IOException* durante il tentativo). Questo, ad esempio, avviene quando il dispositivo in uso non riconosce il formato scelto per l'immagine. L'eventualità, ad ogni modo, è remota, giacché il formato PNG è tra i più digeribili per i dispositivi wireless dotati di supporto a J2ME.

2. Quando l'immagine è più grande dello schermo. In questo caso, infatti, non sarebbe idoneo mostrare un logo che esca dai bordi del display.

Fig. 3: **Il gioco "gira" su diversi cellulari.**

In ambo i casi, si rinuncia alla visualizzazione dell'immagine selezionata, e si opta per l'inserimento di una scritta "MySnake" sostitutiva. Completate le operazioni preliminari, quanto elaborato viene centrato sullo schermo, servendosi delle apposite funzionalità offerte dalle API di J2ME (cfr. documentazione ufficiale).

Carlo Pelliccia

Bibliografia

- **J2ME - GUIDA PRATICA ALLA PROGRAMMAZIONE DI DISPOSITIVI WIRELESS** *John W. Muchow* (McGraw-Hill Italia) ISBN: 88-386-4278-8
- **J2ME IN A NUTSHELL** *Kim Topley* (O'Reilly) ISBN: 0-596-00253-X

Macromedia COMUNICATION SERVER

PARTE SECONDA

Flash

File sul CD
soft\codice
\Videosorveglianza.zip

Codice
Contrariamente a quanto fatto per la videoconferenza, nella quale avevamo implementato le funzioni che ci interessavano in maniera visuale, senza scrivere cioè neanche una riga di codice, stavolta la nostra applicazione sarà basata quasi esclusivamente sulla stesura di un listato che ci permetta di esaudire i requisiti preposti.

Lo scorso mese abbiamo potuto vedere come sia facile, utilizzando gli ultimi prodotti partoriti dalla Macromedia e dedicati al lato server di applicazioni distribuite, integrare nei propri progetti soluzioni potenti e configurabili per la realizzazione di contenuti multimediali. Il sistema di videoconferenza sviluppato nel precedente articolo dovrebbe avervi rese chiare le potenzialità di questi strumenti che riescono, con l'ausilio di pochi e semplici comandi, a generare moduli software completi e perfettamente integrabili in contesti multi-tier. Ovviamente l'esempio introdotto la volta precedente non era che uno dei tanti possibili, quindi andiamo questa volta a prendere in considerazione un altro possibile impiego del server Remote Communication della Macromedia.

Tornate in ufficio come ogni mattina e, come ogni mattina, vi accorgete che i fogli di appunti che avevate lasciato sulla scrivania sono stati spostati, il telefono è stato riappeso male e la vostra spillatrice preferita è sparita… ma chi mai si divertirà a frugare il naso nelle vostre cose mentre non ci siete? Dopo i primi 5 minuti di ira piano piano cominciate a riprendere la lucidità mentale caratteristica di un programmatore del vostro calibro e lentamente mettete a fuoco la situazione. Finalmente, dopo qualche istante, nella vostra mente è tutto chiaro ed un ghigno satanico vi si dipinge in volto mentre sussurrate la parola *"Communication…"*. Già, *Communication*, lo stesso prodotto che il mese scorso vi ha permesso di fare incetta degli elogi del vostro capo, stavolta può aiutarvi ad uscire da questa antipatica situazione, facendovi scoprire chi è che si intrufola nella stanza in vostra assenza. Come può un software arrivare a tanto? Molto semplicemente, registrando dalla vostra webcam tutto ciò che accade quando voi non ci siete! Ma visto che le ore nelle quali non siete nella vostra stanza sono parecchie, sarebbe impensabile registrare ininterrottamente per tutto quel tempo, senza pensare poi alla noia del doversi rivedere ore di immagini, fino al punto in cui effettivamente succede qualcosa. Proprio per non riempire il proprio HardDisk con GigaByte di filmati inutili e non perdere tempo a spulciare gli stessi alla ricerca di qualcosa di interessante, svilupperemo una piccola applicazione che ci permetta di:

- Iniziare a riprendere dalla nostra webcam quando qualcosa si muove nella stanza.
- Terminare la ripresa dopo un certo periodo di tempo dalla cessazione del movimento.
- Rivedere i filmati registrati in maniera semplice e veloce.

Questo ci permetterà di registrare sul nostro computer esclusivamente i momenti nei quali è in corso una qualche azione nell'area ripresa dalla nostra webcam, e di limitare quindi la visione dei filmati alle sole scene fondamentali. Vediamo quindi come fare per procedere nello sviluppo di un'applicazione di questo tipo.

PICCOLO RIASSUNTO

Ricordo che i software necessari per far girare l'applicazione sviluppata in questo articolo, cioè *FlashMX*, *Communication server* ed i *Communication Components*, sono reperibili nelle loro versioni dimostrative agli indirizzi: *www.macromedia.com/it/software/trial_download/* e *www.macromedia.com/software/flashcom/download/components/*. Prima di introdurre l'esempio che andremo a sviluppare, mi preme riassumere brevemente a coloro che non avessero seguito il precedente articolo, i passaggi necessari all'installazione ed alla configurazione di *Remote Communication*.

- In fase di installazione di Communication varrà richiesto una *User-Id* e *password* necessarie poi successivamente per accedere al programma.
- Sempre durante l'installazione, quando vi sarà richiesto, scegliete di utilizzare il prodotto come sviluppatori e non come semplici operatori.
- Ancora durante l'installazione di *Communication*, se avete un altro web-server sul vostro computer, potete poi scegliere se affiancare il prodotto in questione a quello già presente, sebbene ciò non

sia necessario ai fini dell'esempio riportato nell'articolo.

- Per installare i componenti, copiare il file *"Communication Components.fla"* nella cartella *\First Run\Components* situata nella cartella di installazione di FlashMX.
- Copiare la cartella *\scriptlib* presente nel pacchetto dei componenti scaricato, nella directory *\flashcom* risiedente nella directory di installazione di Communicator, sovrascrivendo quella presente.
- Per fare in modo che Communication "veda" l'applicazione che stiamo per realizzare, è necessario creare una cartella con il nome desiderato per l'applicazione stessa, nella directory *\flashcom\applications*, che potete trovare dentro il percorso di installazione che avete scelto per *Communicator.*
- Create un file di testo contenente la stringa *"load ("component.asc");"* e salvatelo con il nome *"main .asc"* nella cartella appena creata.
- Nella stessa cartella create in FlashMX il file *.fla* con lo stesso nome della cartella che conterrà la nostra applicazione.

SI COMINCIA

Contrariamente a quanto fatto per la videoconferenza, nella quale avevamo implementato le funzioni che ci interessavano in maniera visuale, senza scrivere cioè neanche una riga di codice, stavolta la nostra applicazione sarà basata quasi esclusivamente sulla stesura di un listato che ci permetta di esaudire i requisiti preposti. In questo modo potremo vedere come spingerci un po' più in là nella programmazione del prodotto, in modo da poter sfruttare capacità altrimenti non raggiungibili con il semplice settaggio di qualche proprietà ai componenti presenti sullo stage. Ciononostante, i primi passaggi che affronteremo riguarderanno proprio il piazzare sullo stage gli elementi che caratterizzeranno la nostra applicazione. A tal scopo, una volta aperto FlashMX, visualizzate il pannello Libreria dal menu *Window\Library*, e fate clic sulla piccola icona presente sulla destra della sua barra. Dal menù che vi si presenterà, scegliete *New Video* e la vostra libreria si popolerà di un nuovo simbolo. Trascinate dalla libreria il simbolo sullo stage ed avrete un'istanza utilizzabile del simbolo stesso. A cosa ci servirà questa istanza? Semplice, a visualizzare i filmati che saranno registrati dalla nostra webcam. Nella nostra applicazione infatti, appena la registrazione avrà inizio, a schermo saranno mostrate le immagini in fase di salvataggio. Appena trascinata questa istanza sullo stage, il nostro prossimo compito sarà quello di darle un nome in modo da poterci riferire a lei in maniera univoca. Per fare ciò, aprite il pannello *Proprietà* dal menu *Windows\Properties* e, dove appare la scritta *<Instance Name>*, digitate il nome *"schermo"*. Fatto ciò, possiamo passare immediatamente ad immettere il listato che si occuperà della gestione di tutto il lavoro della nostra applicazione. Selezionate il primo fotogramma presente sulla linea temporale nel pannello *Timeline*, dopo di che aprite il pannello *Azioni* dal menu *Windows\Actions* e digitate dentro di esso il codice seguente:

Fig. 1: Il pannello libreria popolato con il nuovo simbolo.

Fig. 2: Il pannello proprietà con il nome dell'istanza.

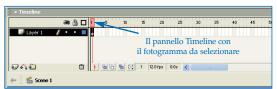



Fig. 3: Il pannello timeline con il fotogramma selezionato.

```
// Imposta il filmato corrente
filmato=0;
// Crea l'oggetto connection necessario ad impostare
                          la comunicazione con il server
connection = new NetConnection();
// Fa puntare l'oggetto al file che deve occuparsi della
                                              connessione
connection.connect("rtmp://localhost/Controllo");
// Crea l'oggetto che si occuperà dello streaming video
publishStream = new NetStream(connection);
// Crea l'oggetto che rappresenta la nostra webcam
myCam = Camera.get();
// Poi ne imposta alcune proprietà
myCam.setQuality(0,100);
myCam.setMode(400,400,30);
mycam.setMotionLevel(50, 5000);
// Infine imposta l'evento di attivazione.
myCam.onActivity = function(mode){ registra(mode);};
// La funzione che si occupa dell' inizio e fine della
                                           registrazione.
function registra(mode){
  // Se mode è true allora inizia a registrare.
  if(mode){
```

Flash

Macromedia *Communication Server*

Componenti

La macromedia rilascia in continuazione nuovi componenti per i suoi prodotti. Visitate spesso il sito per essere aggiornati sugli ultimi sviluppi e sugli ultimi prodotti.

www.macromedia.it

Flash

Macromedia Communication Server

```
        // Fa in modo che a schermo si vedano le immagini
                                 che la webcam sta riprendendo
        schermo.attachVideo(myCam);
        // Ora crea sul server un file con la registrazione
                                        delle immagini riprese.
        publishStream.publish("video"+filmato, "record");
        // Ed infine vi collega ciò che la webcam sta
                                                  riprendendo.
        publishStream.attachVideo(myCam);
    }
    // Se mode è false allora termina la registrazione.
    else{
        // Chiude il flusso in output.
        publishStream.close();
        filmato += 1;
    }
}
// Infine ferma il ciclo del fimato.
stop();
```

Come è possibile vedere anche in questo caso, molto lavoro è stato fatto in casa Macromedia per rendere lo sviluppo di applicazioni multimediali avanzate, semplice ed immediato per tutti. Con poche righe di codice siamo riusciti a realizzare un modulo software completo che ci permette di compiere operazioni fino a qualche tempo fa proibitive per chi si era affacciato da poco al mondo della programmazione. Ma scendiamo nel dettaglio ed andiamo ad osservare il codice da vicino. Il listato inizia dichiarando alcune variabili, che corrispondono ad altrettanti oggetti built-in dello strumento, che ci torneranno utili in seguito. La variabile filmato, la prima che incontriamo, tiene il conto del filmato che stiamo registrando. La nostra applicazione infatti inizia una nuova registrazione ogni qual volta c'è del movimento davanti alla webcam, e si interrompe quando il movimento cessa. Sebbene sia possibile accodare tutte le registrazioni in un unico file sul server (e nella maggior parte dei casi sarebbe più conveniente) , noi faremo in modo di creare ogni volta un filmato diverso, in modo da poter controllare nelle proprietà del file, l'orario di registrazione dello stesso, e sapere a che ora si è verificata l'intrusione nella nostra stanza. La seconda variabile che dichiariamo contiene un oggetto *NetConnection*, necessario per effettuare la connessione con l'applicazione sul server. Proprio per poter effettuare la connessione, dopo aver dichiarato la variabile dobbiamo collegarla all'indirizzo dove la nostra applicazione risiede utilizzando la funzione *"connect("rtmp://localhost/Controllo");"*. Analizzando la stringa, *rtmp* identifica il protocollo di comunicazione, *localhost* il percorso dove risiede il server Communication e *Controllo* altri non è che il nome della cartella che abbiamo creato nella directory *\flashcom\applications* del server e nella quale abbiamo salvato il file Flash omonimo sul quale stiamo lavorando. Il passaggio successivo consiste nel dichiarare un oggetto *NetStream* che si occuperà, sfruttando la connessione appena realizzata, del passaggio dei contenuti in *streaming* da client a server. Infine è necessario creare un ultimo oggetto che rappresenti la nostra webcam, operazione che si ottiene richiamando la funzione *get* dell'oggetto *Camera*. Avendo gli strumenti con cui lavorare, possiamo ora dedicarci alle azioni necessarie per far tenere all'applicazione il comportamento desiderato. Iniziamo, subito dopo le dichiarazioni, a settare alcune proprietà dell'oggetto *Camera* per impostarne la qualità, il modo e la soglia di attivazione, e subito dopo ridefiniamo il metodo *onActivity* dell'oggetto. Questo metodo è di fondamentale importanza per il nostro software in quanto è quello che viene richiamato dalla webcam in caso essa noti del movimento. Il parametro passato al metodo è un booleano che si occupa di rendere noto se il movimento è iniziato *(true)* o terminato *(false)*, ed in base a ciò ci permette di iniziare o terminare la registrazione. Per farlo, si appoggia alla funzione che segue, che si occupa di verificare se è stato notato del movimento o meno e si comporta di conseguenza. In caso la webcam abbia notato del movimento, la funzione fa partire la proiezione a schermo delle immagini in registrazione, poi crea sul server un file di nome "video" più il numero di filmato al quale siamo arrivati, ed infine riempie il file appena creato con i contenuti ripresi dalla webcam. Nel caso invece la funzione sia stata richiamata per indicare la fine del movimento, allora essa altro non fa che chiudere lo stream di collegamento e quindi di conseguenza il file sul server, ed aggiornare la conta dei filmati. L'unica altra operazione necessaria consiste nel fermare il filmato corrente con l'istruzione *stop* che evita che le istruzioni immesse vengano eseguite in un loop continuo. E questo è tutto ciò di cui abbiamo bisogno per tenere sotto controllo la nostra scrivania. Ma manca ancora qualcosina. In effetti possiamo si registrare correttamente qualsiasi movimento notato dalla webcam, ma non abbiamo ancora nessuno strumento per poter rivedere i filmati registrati. Ecco perché ora andremo a realizzare un qualcosa che ci permetta di proiettare i filmati registrati dalla nostra applicazione sul server web.

RIVEDERE I FILMATI

Come ho già spiegato, il nostro software si occupa di registrare una serie indefinita di filmati sul server che

Fig. 4: **La webcam mentre registra.**

Approfondimenti

Sul sito della macromedia troverete importanti informazioni nonché tutorial e reference per imparare ad utilizzare meglio Communication e FlashMX

www.macromedia.it

Flash

Macromedia
Communication Server

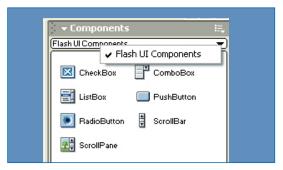



Fig. 5: Il pannello Componenti con il set selezionato.

riprendono ciò che accade davanti alla nostra webcam quando effettivamente accade qualcosa. Visto che non sappiamo quindi a priori la quantità di filmati che verranno registrati e visto che non esiste un metodo semplice per ricavare questo dato da dentro la nostra applicazione, per semplificare l'esempio sarà necessaria un'operazione manuale in fase di visualizzazione delle scene registrate. Ma vediamo come iniziare l'operazione. Prima di tutto aprite il pannello *Componenti da Windows/Components* e selezionate dall'elenco a discesa il set *Flash UI Components* come in Fig. 5. Appena fatto ciò trascinate in un punto libero dello stage un'istanza del componente *ListBox* ed assegnatele il nome *"lista"* dal pannello *Proprietà* come fatto in precedenza per il Video. Bene, ora abbiamo un elemento dal quale selezionare il filmato che ci interessa vedere. Ora aprite di nuovo il fotogramma nel quale avevate immesso del codice precedentemente e, prima di qualsiasi altra istruzione, scrivete:

```
NUMERO_FILMATI = 50;
// Riempie la lista di dati.
for(i=1;i< NUMERO_FILMATI;i++)
   {lista.addItem("video"+i);}
```

Il codice riportato altro non fa che popolare il componente *ListBox* con 50 elementi in ordinati che corrispondono ad altrettanti potenziali filmati. Visto che non sappiamo il numero preciso dei filmati che registreremo, occorre di volta in volta impostare la costante *NUMERO_FILMATI* al giusto valore. Per conoscere il numero esatto dei filmati che sono stati registrati, occorre accedere alla cartella *streams/_definst_* che risiede nella cartella della nostra applicazione e vederene il numero esatto aiutandoci con il loro nome. Ovviamente sarebbero possibili più modi per rendere automatica questa operazione, ma questo comporterebbe la trattazione di argomenti estranei all'articolo, per cui, per stavolta, ci accontenteremo di questa soluzione poco "pulita". Più facile e comune sarebbe invece stato accodare in un unico filmato tutte le riprese, in modo da richiamare quest'unico file senza doverci porre il problema del numero di registrazioni presenti sul server ma, come già spiegato, nel nostro caso potrebbe risultare utile sapere l'orario di registrazione di una determinata scena e ciò è possibile solamente controllando le proprietà di ogni singolo file. Avendo ora a disposizione la lista dei filmati che abbiamo registrato, occorre un evento che ci permetta, una volta selezionatone uno, di avviare la riproduzione. Abbiamo quindi bisogno di un pulsante al quale associare l'evento. Create quindi un pulsante o importatene uno da una libreria condivisa (*Windows-Common Libraries-Buttons*) e posizionatelo sullo stage, poi selezionatelo ciccandovi sopra ed aprite il pannello azioni, dopodiché scrivete il codice riportato:

```
on (release) {
        numVideo=lista.getSelectedItem().label;
        newStream = new NetStream(connection);
        newStream.play(numVideo);
        schermo.clear();
        schermo.attachVideo(newStream);}
```

Analizzando il listato riportato appare subito evidente che è associato all'evento di rilascio dopo la pressione del tasto in questione. Innanzitutto la funzione estrae il nome del filmato selezionato nella lista, che corrisponde al nome di uno dei file registrati sul server, e memorizza il dato in una variabile. La fase successiva consiste nel creare un nuovo oggetto *NetStream* che si occupi del passaggio dei dati in streaming tra client e server, ma stavolta in senso inverso. Si dice infatti nell'istruzione successiva, di mandare in riproduzione il filmato e successivamente si collega la riproduzione allo schermo presente sullo stage. Tutto qui. Questo basta affinché il video venga riprodotto sul nostro computer e la nostra spillatrice ritorni al proprio posto. Quello che serve ora è provare l'applicazione. Ricordatevi di pubblicare il tutto selezionando *File/Publish* o premendo contemporaneamente *Shift* ed *F12*. Fatto questo, lanciate il file *.html* che troverete nella cartella dove avete salvato l'applicazione e controllate che la registrazione si attivi se qualcosa viene mosso davanti alla webcam. Fatto ciò, controllate nella sottocartella *streams/_definst_* che effettivamente qualcosa sia stato registrato ed impostate il numero di filmati dentro Flash. Bene, ora siete in grado di visualizzare le registrazioni semplicemente selezionandole nella lista e premendo il pulsante, ma ricordatevi di indirizzare la webcam verso qualcosa di statico, altrimenti essa verrà attivata di nuovo.

CONCLUSIONI

Ecco svelato un altro possibile utilizzo di questo prodotto che da pochissimo si è affacciato sulla scena informatica. L'esempio di questo articolo è veramente essenziale, ma esso serve esclusivamente ad indirizzarvi verso le capacità offerte da questi nuovi software a chi intenda realizzare applicazioni o siti ricchi di contenuti avanzati. A voi il compito di esplorarne le potenzialità.

Giuliano Uboldi

Multimedia

Molto lavoro è stato fatto in casa Macromedia per rendere lo sviluppo di applicazioni multimediali avanzate, semplice ed immediato per tutti. Con poche righe di codice siamo riusciti a realizzare un modulo software completo che ci permette di compiere operazioni fino a qualche tempo fa proibitive per chi si era affacciato da poco al mondo della programmazione.

Sicurezza

TEORIA E REALIZZAZIONE DI UNO SNIFFER

Sicurezza

Sbirciare tutto ciò che circola su una rete LAN non è un'utopia. Con l'aiuto di Winsock2 e dei raw sockets, scopriamo come si fa a progettare un semplice packet sniffer, strumento molto conosciuto dagli amministratori di rete e dagli hackers!

File sul CD
\soft\codice\SPS.CPP

Sul Web

Packet Sniffer Commerciali e Freeware
http://lists.gpick.com/pages/Packet_Sniffers~Info.htm

Progetto di packet capture driver realizzato dal Politecnico di Torino
http://winpcap.polito.it/

Documento polemico contro i raw sockets redatto dell'esperto Steve Gibson
http://grc.com/dos/sockettome.htm

Dettagli sull'uso della primitiva WSAioctl() di Winsock2
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winsock/winsock/wsaioctl_2.asp

Chi ha visto qualche film di spionaggio di qualche decennio fa, ricorderà sicuramente le scene in cui gli investigatori dell'FBI intercettavano le telefonate dei gangster ricorrendo al vecchio trucco di allacciarsi con degli spinotti lungo il tracciato telefonico. Questo vecchio stratagemma, permetteva agli agenti di spiare e ascoltare tutto ciò che veniva detto su una linea telefonica, conoscendo così in anticipo le mosse dei "cattivi". Son passati parecchi anni da allora e forse ci sembrerà anacronistico oggi scoprire che certi vecchi sistemi non invecchiano mai e a volte mutano, riproponendosi ai giorni nostri sotto nuove spoglie! Pensiamo infatti a come vengono scambiate le informazioni lungo una rete informatica….Oggi, in un certo senso è cambiata la forma di ciò che circola lungo i cavi di telecomunicazioni (bit, non più parole!) e le nuove tecnologie rendono impossibile ad un operatore umano di collegarsi direttamente ad un filo per "ascoltare" cosa viene detto. Tuttavia le reti oggi, seguono principi di funzionamento nati parecchi fa, quando la sicurezza delle informazioni scambiate non era ancora un requisito fondamentale. Ciò ha portato alla nascita nuovi tipi di attacchi e nuove tecniche di spionaggio, che nel caso delle reti informatiche si basano su particolari programmi conosciuti come "sniffer".

PACKET SNIFFER

Un packet sniffer è un programma che, come suggerisce l'onomatopea del suo nomignolo, permette di "annusare" i bytes che circolano su una rete locale. Si tratta di un software molto particolare, nato in passato sui sistemi Linux e che negli ultimi anni ha trovato la sua evoluzione naturale anche su piattaforma Windows, a causa della continua espansione delle reti.

Definire in modo preciso uno sniffer non è semplice…. Esso può di fatto considerarsi una via di mezzo tra un firewall e un debugger, perché si inserisce fra il sistema operativo e la connessione di rete (proprio come un firewall) per catturare i pacchetti scambiati, byte dopo byte, ed allo stesso tempo è in grado di analizzarli e decodificarli (particolare che lo avvicina molto ad un debugger), permettendo di cercare errori di trasmissione o risalire alle informazioni sullo stato di una rete. L'arduo compito che ci proponiamo oggi in questo articolo, è proprio quello di progettare un semplice e rudimentale sniffer in grado di funzionare sotto Windows 2000/XP e capace di intercettare tutto il traffico IP di una rete locale, decodificando i campi di alcuni dei protocolli comunemente usati (TCP, UDP, ICMP). I requisiti per la realizzazione del nostro "sniffer" saranno le nozioni di base sul funzionamento delle reti e sulla struttura dei pacchetti TCP/IP, unite ad una buona conoscenza del linguaggio C++ e al concetto dei Raw Socket, un nuovo formidabile strumento messo a disposizione dei programmatori da Windows 2000 e XP. Alcune delle nozioni fondamentali saranno trattate, in modo sintetico, nella parte introduttiva dell'articolo, prima di arrivare alla presentazione del codice C++ vero e proprio.

COME FUNZIONA UNO SNIFFER?

Il funzionamento di uno sniffer è basato sostanzialmente sul modo in cui lavorano le reti e su come vengono scambiate le informazioni (pacchetti) fra computer. Lo standard Ethernet venne progettato

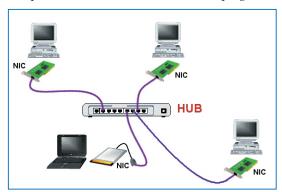



Fig. 1: Schema di una generica LAN

Sicurezza

Sicurezza
Teoria e realizzazione di uno sniffer

molto tempo fa sulla base di un principio molto importante, il "medium share", cioè la condivisione del mezzo di trasmissione. Tutti i PC connessi ad una stessa rete condividono infatti il medesimo cavo, che devono usare alternativamente per inviare e ricevere dati. Questa considerazione implica un fatto importante: tutti gli host sono spettatori passivi del traffico scambiato sul cavo condiviso e sono quindi potenzialmente in grado di vedere i pacchetti che circolano sulla rete, anche quando non sono esplicitamente di loro proprietà! Un pacchetto spedito da un computer viaggia lungo la rete locale fino all'HUB, quella periferica che svolge il compito di mettere in comunicazione i diversi host di una rete (nodo accentratore). L'HUB lavora in maniera abbastanza stupida: non fa altro che duplicare e rispedire il pacchetto ricevuto lungo tutti i cavi (broadcast) connessi alle sue porte, eccetto quella dalla quale il pacchetto è arrivato, sperando che attraverso una delle porte, il pacchetto giungerà al destinatario.

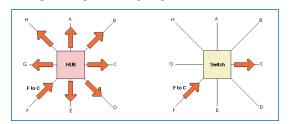



Fig. 2: Schema di funzionamento hub/swtich

Questo fatto ha un'implicazione abbastanza rilevante: ogni pacchetto che transita attraverso un HUB, viene di fatto spedito a tutti gli host di una rete locale, anche quando questo non è strettamente necessario! Ma cosa succede allora quando il pacchetto giunge al capolinea? Al termine del suo cammino (se non è andato perso) il pacchetto sarà di sicuro ricevuto e processato da una qualche interfaccia di rete (in breve NIC = Network Interface Card) di un computer, che si comporta come un vero e proprio "filtro" hardware: dal pacchetto vengono estratti l'indirizzo del mittente e del destinatario e se quest'ultimo differisce dall'indirizzo fisico dell'interfaccia di rete, allora l'intero pacchetto viene scartato. La scheda di rete quindi elabora un pacchetto (e lo passa al sistema operativo, ad un livello di rete superiore) solo quando questo è effettivamente destinato al computer in cui essa è inserita. Il resto dei pacchetti superflui, giunge comunque alla scheda ma viene immediatamente scartato. Guardando le cose ad un livello più approfondito, lo schema di incapsulamento di un generico pacchetto Ethernet (a meno del preambolo iniziale) appare alla scheda di rete in questo modo:

ETHERNET FRAME				
Dst	Src	Type	Data...	FCS
<-- 6 -->	<-- 6 -->	<-- 2 -->	<-46…1500->	<- 4 ->BYTES

I primi 2 campi (di 6 bytes ciascuno) sono gli indirizzi fisici del destinatario e del mittente del pacchetto, mentre il campo Type indica il tipo di protocollo di livello superiore contenuto all'interno del frame (ad esempio IPv4 ha il codice 0800). Seguono i dati, la cui lunghezza non può superare i 1500 bytes ed infine una sorta di checksum di integrità del pacchetto, composto da 4 bytes. La scheda Ethernet lavora come un filtro, maneggiando questo tipo di frame: estrapola le informazioni sul destinatario leggendo i 6 bytes iniziali e li confronta col proprio indirizzo univoco, decidendo se rigettare o processare il pacchetto.

PROMISCUOUS MODE

Fin qui nulla di strano, si tratta più che altro di una ripetizione veloce di alcune tecnologie fondamentali delle reti. Quello che molti non sanno però è questo: esiste una modalità di funzionamento delle schede di rete davvero particolare, chiamata promiscuous mode, nota per lo più ai tecnici di rete, agli amministratori di sistema e ovviamente agli hackers! Quando una scheda di rete lavora in modo promiscuo, in sostanza viene disabilitata l'operazione di filtraggio di cui si è parlato prima e di conseguenza, la scheda inizia a processare e a tenere in considerazione tutti i pacchetti in transito sulla rete (anche quando sono destinati ad altri!), senza scartarne nessuno. Questa funzionalità delle schede di rete, esiste per scopi puramente diagnostici e serve agli amministratori per effettuare analisi sul traffico e sulla congestione da una qualsiasi postazione; tuttavia è anche vero che si tratta di una terribile arma se usata dagli hacker, che in questo modo possono vigilare e catturare tutti i dati in transito su una rete locale (password, numeri di carte di credito e altre informazioni sensibili), restando comodamente seduti sulla loro postazione. Uno sniffer inizia il suo lavoro quindi attivando il promiscuous mode della scheda di rete per catturare i pacchetti in transito, ma non abbiamo ancora specificato su quale computer deve essere eseguito. L'host di esecuzione dello sniffer, per quanto detto prima, non ha importanza, nel senso che l'intercettazione del traffico può avvenire da un qualsiasi punto della rete locale, visto che i pacchetti viaggiano in broadcast a causa del

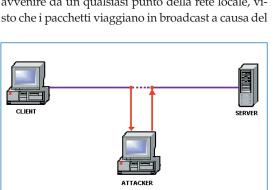



Fig. 3: Man-in-the-middle.

Neutralizzare uno sniffer: SWITCH vs HUB

Lo SWITCH è una periferica di rete più avanzata e più intelligente degli HUB. Si tratta di un dispositivo che svolge sempre un ruolo di interconnessione su una rete locale, ma con la grande differenza – rispetto ad un HUB – che i pacchetti ricevuti vengono smistati sulle porte e sui cavi dove effettivamente sono diretti. Lo SWITCH infatti "studia" la configurazione di una rete dinamicamente e associa gli indirizzi delle schede di rete alle sue porte, evitando il broadcast su tutti i canali, che consente ad uno sniffer di intercettare il traffico.

Sicurezza

Sicurezza
Teoria e realizzazione di uno sniffer

lavoro fatto dagli HUB. Quindi a differenza di ciò che accadeva nei vecchi film di spionaggio, non è più necessario un attacco di tipo "man-in-the-middle", dove l'intercettazione avviene infiltrandosi direttamente sul canale di comunicazione; nel caso dello sniffer, l'attacco può avvenire da una qualsiasi postazione locale della LAN e su reti molto estese (ad esempio un campus universitario con centinaia di punti rete dislocati in posti diversi) diventa un compito difficile capire da dove è stato azionato uno sniffer! Tutto questo ci porta ad un'ultima importante considerazione: un attacco basato su uno sniffer, avverrà sempre da un computer (o da un indirizzo IP) presente all'interno di una rete locale, non è cioè possibile che un hacker possa "sniffare" il nostro traffico locale dall'esterno della rete.

RAW SOCKETS E MICROSOFT

Il concetto di "socket" è ormai un paradigma ben noto a tutti quei programmatori che hanno scritto una qualsiasi applicazione di rete. L'uso dei sockets è abbastanza immediato in tutti i linguaggi, basta specificare un indirizzo IP, le porte da usare e il tipo di protocollo (TCP, UDP) per instaurare una connessione di rete con poche righe di codice. L'astrazione dei sockets è molto potente, perché consente di scrivere applicazioni client/server in modo veloce, dove è compito delle librerie TCP/IP preoccuparsi di instaurare la connessione, realizzare l'handshaking, impacchettare i dati nel modo giusto e spedirli. Tuttavia, guardando la cosa da un punto di vista più tecnico, il meccanismo dei sockets non è completo, perché non ci consente di lavorare sulle reti a un livello più basso. E' impossibile infatti, usando i sockets puri, inviare richieste ICMP ad un altro host (come un ping ad esempio) oppure creare pacchetti con un certo valore di TTL prestabilito. Questo tipo di manipolazioni non sono infatti consentite, perché avvengono a un livello più basso di quello in cui lavorano i socket classici. Per sopperire a tali limitazioni (con l'avvento di Winsock 2) sono stati introdotti nei sistemi operativi Microsoft (in netto ritardo di molti anni rispetto a Linux!) anche i raw socket, uno strumento utilissimo (quanto contestato!) per lavorare ad un livello più basso con le applicazioni di rete. Normalmente infatti, con i sockets tradizionali si opera ai livelli 4 e 5 (Transport e Session) del modello ISO/OSI, mentre le operazioni di manipolazione dei pacchetti descritte poco fa, avvengono al livello 3 (Network), nel cuore dello stack TCP/IP. Le funzionalità dei raw sockets sono state estese a tutte le famiglie di sistemi Windows col kernel a 32-bit (2000 e XP), mentre sui vecchi Windows 98/ME, è presente un set di ridotto di istruzioni, che consente solo la creazione di semplici pacchetti ICMP. Tra le altre cose da segnalare, va anche detto che sotto Windows 2000 l'uso dei raw sockets è limitato al solo utente Administrator, mentre invece, con Windows XP, Microsoft ha deciso di estendere a tutti gli utenti del sistema l'uso di questi "gingilli"… scelta discutibile, che è stata a lungo contestata su Internet dall'esperto di sicurezza Steve Gibson. Discussioni e moralismi a parte, i raw sockets rappresentano per noi i mattoni indispensabili per costruire il nostro sniffer, che appunto dovrà manipolare i pacchetti ad un livello davvero basso.

LE STRUTTURE DATI DELLO SNIFFER

Abbiamo visto che lo sniffer, dopo aver configurato la scheda di rete in modalità promiscua, riceve tutti i frame Ethernet che circolano su una rete locale; in realtà lo sniffer non agisce effettivamente al livello MAC (layer 2 del modello ISO/OSI), perché è compito dell'interfaccia di rete processare i frame Ethernet, estrapolarne il contenuto (il campo Data lungo 1500 byte, per intenderci) e consegnarlo al livello Network (layer 3). Tale campo, nel nostro caso specifico, conterrà un pacchetto costruito secondo le specifiche dell'Internet Protocol (IP) che, a sua volta, trasporterà all'interno un segmento di tipo TCP, UDP o ICMP. Questo strano meccanismo di scatole cinesi, dove ogni pacchetto viene incastrato all'in-

Livelli ISO/OSI

Livello ISO/OSI	Nome	Protocolli TCP/IP	
7	Application	HTTP, FTP, ...	
6	Presentation	Telnet SSH	
5	Session		
4	Transport	TCP	UDP
3	Network	IP ICMP ARP RARP	
2	Data Link	Device driver Interfaccia fisica	
1	Physical		

Fig. 4: ISO / OSI.

Fig. 5: Encapsulation.

Perchè i raw sockets sono contestati?

Qualche tempo fa l'esperto di sicurezza Steve Gibson, pubblicò un lungo documento http://grc.com/dos/sockettome.htm dove contestava a Microsoft l'introduzione dei raw sockets su Windows XP, chiedendo addirittura che venissero banditi dalla versione finale del sistema operativo. Come mai tanto accanimento contro questi oggetti? La spiegazione è semplice: i raw sockets permettono di manipolare a basso livello i pacchetti di rete e sono quindi uno strumento potente nelle mani degli hacker, che possono (ora anche con Windows XP) creare pacchetti malformati capaci di lanciare attacchi DoS. Gibson in realtà avrebbe voluto che i raw sockets fossero disponibili solo per gli amministratori di sistema e non per tutti gli utenti (che effettivamente non necessitano di questi gingilli), limitandone l'uso solo alla versione Professional di Windows XP.

Sicurezza

Sicurezza
Teoria e realizzazione di uno sniffer

terno di un altro, è ben noto nel mondo delle reti e prende il nome di encapsulation.
Il nostro sniffer deve quindi poter gestire questo meccanismo e deve essere in grado di maneggiare tutti i tipi di protocolli, di conseguenza bisognerà innanzitutto dichiarare le opportune strutture dati che modellano formalmente lo stack del TCP/IP in linguaggio C. Partiamo col definire l'IP Header, schematizzato nel seguente modo:

IP Header				
Version 4 bits	IHL 4 bits	TOS 8 bits	Total Length 16 bits	
Identification 16 bits		Flags 3 bits	Fragment Offset 13 bits	
Time to Live 8 bits		Protocol 8 bits	Header Checksum 16 bits	
Source Address 32 bits				
Destination Address 32 bits				
Options			Padding	

Nel sorgente dello sniffer scriveremo quindi una struct contenente ogni campo dell'IP Header, con le dimensioni indicate dal modello. I campi a 8-bit possono essere rappresentati con degli unsigned char, mentre per quelli a 16 e a 32-bit useremo gli unsigned short e gli unsigned long. I campi "anomali" da 3, 4 e 13-bit saranno invece accorpati in modo da ottenere valori esatti (8 – 16 bit) e verranno gestiti nel sorgente con apposite istruzioni di Shift e And logico sui bit.

```
//IPv4 HEADER (20 BYTES)
typedef struct iphdr
{
  unsigned char  VerIHL; // 8-bit - Version (4-bit) + IP
                                       Header Length (4-bit)
  unsigned char  Tos; // 8-bit - Type of Service
  unsigned short Total_len; //16-bit - Total Length
  unsigned short ID; //16-bit - Identification
  unsigned short Flags_and_Frags; //16-bit - Flags
                           (3-bit) + Fragment offset (13-bit)
  unsigned char  TTL; // 8-bit - Time To Live
  unsigned char  Protocol; // 8-bit - Protocol
  unsigned short Checksum; //16-bit - Header Checksum
  unsigned long  SrcIP; //32-bit - Source IP Address
  unsigned long  DstIP; //32-bit - Destination IP Address
  //unsigned long Opt_and_Padd;
} IpHeader;
```

La dimensione dell'IP Header (escludendo i campi opzionali *Options + Padding*) è di 20 byte. Il datagramma IP possiede campi fondamentali per lo sniffer, come gli indirizzi IP del mittente *(SrcIP)* e del destinatario *(DstIP)* sotto forma di word da 32-bit e il campo Protocol, indispensabile per stabilire se il pacchetto trasportato è di tipo TCP, UDP o ICMP (protocolli identificati rispettivamente dai valori 6, 17 e 1). Per una spiegazione dettagliata del significato di tutti i campi, rimandiamo ai numeri 60 e 61 di ioProgrammo.
Discorso simile va fatto anche per gli header dei protocolli TCP, UDP e ICMP, che andranno definiti con altrettante struct all'interno del nostro sorgente.

TCP Header			
Source Port 16 bits		Destination Port 16 bits	
Sequence Number 32 bits			
Acknowledgment Number 32 bits			
D-Offset 4 bits	Reserved 6 bits	Ctrl Bits 6 bits	Window 16 bits
Checksum 16 bits		Urgent Pointer 16 bits	
Options		Padding	
Data			

```
//TCP HEADER (20 BYTES + DATA)
typedef struct tcphd
{
  unsigned short SrcPort; //16-bit - Source Port
  unsigned short DstPort; //16-bit - Destination Port
  unsigned long  SeqNumber; //32-bit - Seq Number
  unsigned long  AckNumber; //32-bit - Ack Number
  unsigned short Off_Res_Ctrl; //16-bit - D-Offset
                 (4-bit) + Reserved (6-bit) + Ctrl (6-bit)
  unsigned short Window; //16-bit - Window
  unsigned short Checksum;   //16-bit - Checksum
  unsigned short Urgent; //16-bit - Urgent Pointer
  //unsigned long Opt_and_Padd;
  unsigned char dati[MAX_PACKET_SIZE];
} TcpHeader;
```

UDP Header	
Source Port 16 bits	Destination Port 16 bits
Length 16 bits	Checksum 16 bits
Data	

```
//UDP HEADER (8 BYTES + DATA)
typedef struct udphd
{
  unsigned short SrcPort; //16-bit - Source Port
  unsigned short DstPort; //16-bit - Destination Port
  unsigned short Length; //16-bit - Datagram Length
  unsigned short Checksum; //16-bit - Checksum
  unsigned char dati[MAX_PACKET_SIZE];
} UdpHeader;
```

Gli header TCP e UDP sono seguiti da un campo dati (definito come array di char), che contiene i byte informativi veri e propri trasportati nel pacchetto; poiché la lunghezza massima di un pacchetto IP è espressa da un campo a 16-bit (Total Length), avremo al massimo pacchetti grandi 2^{16} = 65.536 byte per

I misteri di SIO_RCVALL su Windows XP

Per qualche strano motivo (o per un bug), pare che il promiscuos mode attivato con SIO_RCVALL su Windows XP non funzioni proprio perfettamente. Lo sniffer in azione sotto i sistemi XP cattura infatti tutti i pacchetti della rete (anche quelli altrui), ma non quelli uscenti dall'host su cui viene eseguito lo sniffer...che in teoria dovrebbero essere immediatamente visibili! Sui forum riservati ai developer di tutto il mondo la questione riguardo questo comportamento di SIO_RCVALL è ancora aperta. Ma la stranezza più grande è questa: attivando il firewall incluso con Windows XP, misteriosamente lo sniffer inizia a catturare anche i pacchetti in uscita...cosa avrà combinato questa volta Microsoft?

Sicurezza

Sicurezza
Teoria e realizzazione di uno sniffer

ciascun datagramma IP, da cui bisogna togliere i 20 byte dell'intestazione. In realtà però, se lo sniffer agisce su una rete Ethernet, non incontrerà mai pacchetti più grandi di 1500 bytes, perché la MTU di queste reti è fissata appunto su questo valore (entrano in gioco le problematiche della frammentazione di cui non ci occuperemo). Il valore MAX_PACKET_SIZE è quindi una costante scelta dal programmatore, definita nella sezione iniziale del sorgente (SPS.CPP), assieme ad altre costanti e variabili di uso globale.

```
//SPS.cpp
//Simple Packet Sniffer (SPS)
//(c) Elia Florio (eflorio@edmaster.it)
#include <stdio.h>

//WINSOCK 2 - RAW SOCKET SUPPORT
#include <winsock2.h>
#pragma comment (lib, "ws2_32.lib")
#define SIO_RCVALL _WSAIOW(IOC_VENDOR,1)
                               //required by winsock2.h
#define MAX_PACKET_SIZE (65536-20)
                               //2^16 - ip_header_length

//log file C:\SPSLOG.TXT
FILE *file=NULL;
char* filename;

//menu' variables
bool filelog;
bool hexlog;
char ipaddr[255];
int  filter_type;
```

```
"D:\AT_WORK\ioProg_PacketSniffer\SPS\Debug\SPS.ex
SIMPLE PACKET SNIFFER
---------------------
Written by Elia Florio
(eflorio@edmaster.it)

Local IP Adrress  ? 10.0.0.1

1> Log immediato solo su console
2> Log su console e su file (C:\SPSLOG.TXT)
Modalita' di logging ? 2

1> Dump in formato ASCII leggibile
2> Dump in formato HEX
Modalita' di dump ? 2

1> TCP
2> UDP
3> ICMP
4> TCP / UDP / ICMP
Pacchetti da analizzare ? 4
```

Fig. 6: Le impostazioni dello sniffer.

Sono fondamentali, in questo pezzo di codice d'intestazione dello sniffer, l'inclusione di winsock2.h e della relativa libreria di linkaggio *ws2_32.lib*, specificata mediante la direttiva *#pragma comment*; senza la presenza di queste dichiarazioni, il sorgente *SPS .CPP* non potrà essere compilato.
Altra definizione importante è SIO_RCVALL, richiesta (approfondiremo in seguito) da Winsock per attivare il promiscuous mode della scheda di rete con i raw sockets. Completa infine il quadro delle strutture dati la struct relativa all'header ICMP, definita in questo modo:

```
// ICMP Header (8 BYTES)
typedef struct icmphd
{
  unsigned char  icmp_type; //  8-bit - Type of message
  unsigned char  icmp_code; //  8-bit - Code
  unsigned short icmp_cksum; // 16-bit - Checksum
  unsigned short icmp_id; // 16-bit - Identifer
  unsigned short icmp_seq; // 16-bit - Sequence number
} IcmpHeader;
```

ICMP Header		
Type 8 bits	Code 8 bits	Checksum 16 bits
Identifier 16 bits	Sequenze 16 bits	Number

RAW SOCKETS IN AZIONE

Il corpo dello sniffer è tutto scritto nel main, che si preoccupa di creare un raw socket, eseguirne il binding con l'indirizzo IP della scheda di rete (in modo promiscuo, s'intende) e procedere poi richiamando un'apposita routine di *Sniffing()*, dichiarata a parte. Come si evince dal codice che segue, lo sniffer necessita di due chiamate particolari, rispettivamente alle funzioni *WSAStartup()* e *WSACleanup()*, che servono ad inizializzare (e a terminare) la libreria *WS2_32.DLL* per l'uso dei raw sockets. La chiamata WSAstartup() riceve come parametri una struttura di tipo *WSAData* e una word corrispondente al numero di versione di Windows Socket che si vuole utilizzare (nel nostro caso è sufficiente Winsock 2.1).

```
int main()
{
  WSAData wsaData;
  SOCKET Sock;
  struct sockaddr_in SockAddr;
  DWORD BytesReturned;
  int I = 1;
  int sfres,sfsel;

  try
  {
    if (WSAStartup(MAKEWORD(2, 1), &wsaData) != 0)
    {  printf("Error: WSAStartup() failed\n");
       exit(-1);
    }

    Sock = socket(AF_INET, SOCK_RAW, IPPROTO_IP);

    if (Sock == INVALID_SOCKET)
    {  printf("Error: socket() failed\n");
```

MAC Address

Il MAC Address Ethernet è un numero di 48-bit (6 bytes in tutto) che serve ad identificare univocamente un'interfaccia di rete. Formalmente questo numero si divide in due parti: la prima metà (24-bit) identifica il produttore della scheda, la seconda metà (i restanti 24-bit), sono un serial number (chiamato OUI) assegnato alla scheda. In questo modo ogni scheda di rete è distinguibile in maniera univoca da ogni altra. Schede di rete con indirizzi MAC uguali potrebbero infatti causare collisioni e seri problemi ad una LAN. La tabella con gli OUI assegnati ai produttori è consultabile su http://standards.ieee.org/regauth/oui/index.shtml. Per conoscere il MAC address della propria scheda basta digitare dal prompt il comando "ipconfig /all". L'associazione tra MAC address e indirizzo IP di una scheda di rete, viene fatta mediante una tabella ARP dinamica, consultabile col comando "arp –a".

```
        exit(-1);
    }
```

Il raw socket *(Sock)* viene dichiarato in testa al main ed è istanziato chiamando il costruttore *socket()*, specificando il tipo *(SOCK_RAW)* e il protocollo da usare *(IPPROTO_IP)*. Il passo successivo consiste nel leggere un indirizzo IP da console (inserito manualmente dall'utente che andrà ad usare lo sniffer) ed eseguire il binding del socket su tale indirizzo, memorizzato nel campo *sin_addr* della struct *SockAddr*. L'indirizzo IP deve essere trattato usando la funzione *inet_addr()*, che converte una stringa in un formato di indirizzo di rete valido.

```
memset(&SockAddr, 0, sizeof(SockAddr));
sfres = scanf("%s",&ipaddr);

SockAddr.sin_addr.s_addr = inet_addr(ipaddr);
SockAddr.sin_family = AF_INET;
SockAddr.sin_port = htons(0); //anything...

//binding socket in PROMISCUOUS MODE
if (bind(Sock, (SOCKADDR *)&SockAddr, sizeof(SockAddr))
                                        == SOCKET_ERROR)
{   printf("Error: bind(%s) failed\n", inet_ntoa(
                                    SockAddr.sin_addr));
    exit(-1);
}

if (WSAIoctl(Sock, SIO_RCVALL, &I, sizeof(I), NULL,
                    NULL, &BytesReturned, NULL, NULL)
                                        == SOCKET_ERROR)
{   printf("Error: WSAIoctl() failed\n");
    exit(-1);
}
```

Il cuore di tutto lo sniffer è racchiuso nella chiamata a *WSAIoctl()*, la API che permette di controllare lo stato di un socket. Passando a questo metodo un socket e il parametro *SIO_RCVALL*, è possibile realizzare il promiscuos mode di cui si è a lungo discusso prima.

```
printf("\n1) Log immediato solo su console\n");
printf("2) Log su console e su file (C:\\SPSLOG.TXT)\n");
printf("Modalita' di logging ? ");
sfres = scanf("%d",&sfsel);
if (sfsel==1) filelog=false;
else filelog=true;

printf("\n1) Dump in formato ASCII leggibile\n");
printf("2) Dump in formato HEX\n");
printf("Modalita' di dump ? ");
sfres = scanf("%d",&sfsel);
if (sfsel==1) hexlog=false;
else hexlog=true;

printf("\n1) TCP\n");
printf("2) UDP\n");
printf("3) ICMP\n");
printf("4) TCP / UDP / ICMP\n");
printf("Pacchetti da analizzare ? ");
sfres = scanf("%d",&filter_type);

printf("\n\n\n\n\n------------------BEGIN SESSION\n");

if (filelog) {
   filename="C:\\SPSLOG.TXT";
   file=fopen(filename,"a+");
   fprintf(file,"\n------------------BEGIN SESSION\n");
}

StartSniffing(Sock);
}
```

Per rendere lo sniffer più pratico da usare si è pensato di introdurre due modalità di logging : su console (visualizzando a video i pacchetti intercettati sulla rete) o su file fisico *(C:\SPSLOG.TXT)*. È inoltre utile poter regolare il formato di visualizzazione dei byte catturati, che potranno essere presentati dallo sniffer in formato ASCII puro (comprensibile "a vista" da un utente) o in formato codificato in esadecimale. Si è introdotta infine la possibilità di operare un semplice filtraggio dei pacchetti catturati, fatto in base al tipo di protocollo (TCP, UDP, ICMP). La scelta di questi diverse modalità di funzionamento dello sniffer può essere impostata ad ogni esecuzione dall'utente, mediante semplici scelte da console, presentate prima dell'inizio della cat-

```
"D:\AT_WORK\ioProg_PacketSniffer\SPS\Debug\SI
* [Header IP]
   From: 10.0.0.2     To: 10.0.0.1
   ID: 1556
   TTL: 32768
   Total Length: 60

* [Protocol: ICMP]
   Type: 0      Code: 0
   IcmpSeq: 6
   (EchoReply)

=====CAPTURED PACKET======
* [Header IP]
   From: 10.0.0.2     To: 10.0.0.1
   ID: 1557
   TTL: 32768
   Total Length: 60

* [Protocol: ICMP]
   Type: 0      Code: 0
   IcmpSeq: 7
   (EchoReply)

=====CAPTURED PACKET======
* [Header IP]
```

```
Prompt dei comandi
Microsoft Windows XP [Versione 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

J:\Documents and Settings\pflorik>ping 10.0.0.2

Esecuzione di Ping 10.0.0.2 con 32 byte di dati:

Risposta da 10.0.0.2: byte=32 durata<1ms TTL=128
Risposta da 10.0.0.2: byte=32 durata<1ms TTL=128
Risposta da 10.0.0.2: byte=32 durata<1ms TTL=128
Risposta da 10.0.0.2: byte=32 durata<1ms TTL=128

Statistiche Ping per 10.0.0.2:
    Pacchetti: Trasmessi = 4, Ricevuti = 4, Persi = 0 (0% persi),
Tempo approssimativo percorsi andata/ritorno in millisecondi:
    Minimo = 0ms, Massimo =  0ms, Medio =  0ms

J:\Documents and Settings\pflorik>
```

Fig. 7: **Lo sniffer "a lavoro".**

Sicurezza

Sicurezza
Teoria e realizzazione di uno sniffer

Sniffer

Per rendere lo sniffer più pratico da usare si è pensato di introdurre due modalità di logging:

su console (visualizzando a video i pacchetti intercettati sulla rete);

su file fisico (*C:\SPSLOG .TXT*).

Sicurezza

Sicurezza
Teoria e realizzazione di uno sniffer

tura. Questi menù regolano il valore di alcune variabili globali (*filelog*, *hexlog*, *filter_type*) usate in seguito nella routine *PacketFilter()*. Il main termina con la chiusura del socket e la chiamata finale alla *WSACleanup()*.

```
        catch (...)
        {
            printf("Error: Unknown exception\n");
        }
        closesocket(Sock);
        WSACleanup();

        return 0;
}
```

NEL CUORE DELLO SNIFFER

Come avviene la cattura dei pacchetti? La routine incaricata di questo onere è chiamata non a caso *Sniffing()* e riceve come parametro il raw socket che è stato creato nel main. Ogni singolo datagramma IP in transito sulla scheda di rete viene ricevuto anche dal nostro raw socket e passato alla routine *Sniffing()*, che dovrà quindi definire un buffer capace di contenere un pacchetto *(RecvBuffer)*. È in questa routine che viene impostato il ciclo do-while che riceve di volta in volta tutti i pacchetti in transito. Il buffer viene allocato in memoria come un array di char, con dimensione pari a *MAX_PACKET_SIZE+1*

Pacchetti in transito

Ogni singolo datagramma IP in transito sulla scheda di rete viene ricevuto anche dal nostro raw socket e passato alla routine *Sniffing()*, che dovrà quindi definire un buffer capace di contenere un pacchetto *(RecvBuffer)*. È in questa routine che viene impostato il ciclo *do-while* che riceve di volta in volta tutti i pacchetti in transito.

```
// SNIFFER LOOP
void Sniffing(SOCKET Sock)
{
    char *RecvBuffer = (char *)malloc(
                                MAX_PACKET_SIZE + 1);
    int BytesRecv, FromLen;
    struct sockaddr_in From;
    if (RecvBuffer == NULL)
    {   printf("malloc() failed.\n");
        exit(-1);
    }
     FromLen = sizeof(From);

    //packet-capture loop
do
{
    memset(RecvBuffer, 0, MAX_PACKET_SIZE + 1);
    memset(&From, 0, sizeof(From));
    BytesRecv = recvfrom(Sock, RecvBuffer,
                    MAX_PACKET_SIZE, 0, (sockaddr *)
                                    &From, &FromLen);
   //capture a packet and pass it to filtering function
   if (BytesRecv > 0)
   {PacketFilter(RecvBuffer, BytesRecv);}
   else
   {printf( "recvfrom() failed.\n");}
}
while (BytesRecv > 0);

free(RecvBuffer);
}
```

Ricevere i pacchetti in modo promiscuo è abbastanza immediato con le API di Winsock 2, basta infatti ricorrere alla funzione *recvfrom()*, che accetta come parametri il raw socket *(Sock)*, il buffer dove memorizzare il pacchetto (RecvBuffer), la dimensione massima di tale buffer *(MAX_PACKET_SIZE)*, alcuni flags (settati a 0) e un puntatore ad indirizzo IP, che nel nostro caso è trascurato. Il valore restituito da *recvfrom()* corrisponde al numero di bytes catturati dallo sniffer e viene usato come condizione di continuazione del ciclo nel do-while *(BytesRecv > 0)*. Ogni pacchetto catturato viene immediatamente passato alla routine *PacketFilter()*, che rappresenta l'interprete del nostro sniffer: è quella parte del programma che ha il compito di leggere l'header IP, estrapolare i vari campi dell'intestazione (quelli descritti nel paragrafo precedente), convertirli in un formato leggibile dall'utente e spacchettare i protocolli di livello superiore (TCP, UDP, ICMP) incapsulati nel pacchetto. Ci sarebbe a questo punto un ampio discorso da fare sulla velocità di cattura dei pacchetti rapportata alla capacità di buffering del nostro sniffer, ma si tratta di problematiche che esulano da questo articolo e che richiederebbero una trattazione separata.

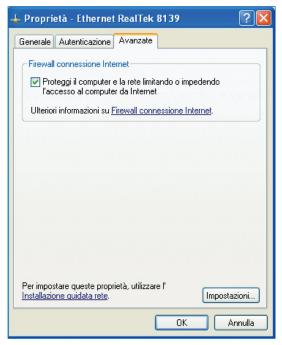



Fig. 8: La finestra di proprietà sulla scheda di rete.

DIMMI CHE PACCHETTO SEI...

PacketFilter() estrae quindi i dati da ogni pacchetto ricevuto e li processa seguendo uno schema ben pre-

Sicurezza

Sicurezza
Teoria e realizzazione di uno sniffer

ciso. È all'interno di questa routine che vengono usate le strutture dati con gli header IP, TCP, UDP e ICMP, che abbiamo visto e definito all'inizio dell'articolo. Per ciascuna di queste struct viene infatti dichiarato un puntatore. La prima cosa da fare per analizzare un pacchetto, è il casting dell'oggetto Buffer – passato come argomento dalla routine *Sniffing()* - ad un oggetto di tipo IpHeader*, che modella tutti i campi del protocollo IP. Una volta fatta questa operazione, per accedere ad un qualsiasi campo è sufficiente usare l'operatore "->". Ad esempio: la versione e la lunghezza dell'header IP sono memorizzate nel campo chiamato "VerIHL", sotto forma di un unico byte (4-bit per la versione + 4-bit per la lunghezza); per accedere a questo campo basta usare *"iphdr->VeIHL"*. La lunghezza esatta dell'header IP, che potrebbe essere maggiore di 20 bytes in presenza del campo Options, è rappresentata in questo caso dagli ultimi 4 bit del byte in questione, che si possono ricavare con uno shift di 4 posizioni mediante l'operatore "<<" del C++. Il tutto si traduce nel seguente codice:

```
void PacketFilter(char* Buffer, int Size)
{
        IpHeader *iphdr;
        TcpHeader *tcphd;
        UdpHeader *udphd;
        IcmpHeader *icmphd;
        struct sockaddr_in SockAddr;
        unsigned short iphdrlen;
        char C;
        //Analyze packet
        iphdr = (IpHeader *)Buffer;
        iphdrlen = (iphdr->VerIHL << 4);
        memcpy(&C, &iphdrlen, 1);
        iphdrlen = (C >> 4) * 4; //20 bytes
```

Il resto della routine *PacketFilter()* estrae i campi del datagramma IP *(SrcIP, DstIP, ID, TTL, Total_len)* e li scrive poi sulla finestra della console ed eventualmente (se il valore di *filelog* è *true*) anche sul file *C:\SPSLOG.TXT*.

```
 //Analyze IP Header
 printf("\n===CAPTURED PACKET====\n");
 if (filelog) fprintf(file,"\n===CAPTURED
                              PACKET====\n");

 printf("* [Header IP] \n");
 if (filelog) fprintf(file,"* [Header IP] \n");

 memset(&SockAddr, 0, sizeof(SockAddr));
 SockAddr.sin_addr.s_addr = iphdr->SrcIP;
 printf("  From: %s ", inet_ntoa(SockAddr.sin_addr));
 if (filelog) fprintf(file,"  From: %s ", inet_ntoa(
                              SockAddr.sin_addr));

 memset(&SockAddr, 0, sizeof(SockAddr));
 SockAddr.sin_addr.s_addr = iphdr->DstIP;
 printf("  To: %s \n", inet_ntoa(SockAddr.sin_addr));
 if (filelog) fprintf(file,"  To: %s \n", inet_ntoa(
                              SockAddr.sin_addr));

 printf("  ID: %i \n", ntohs(iphdr->ID));
 if (filelog) fprintf(file,"  ID: %i \n", ntohs(iphdr->ID));

 printf("  TTL: %i \n", ntohs(iphdr->TTL));
 if (filelog) fprintf(file,"  TTL: %i \n", ntohs(iphdr->TTL));

 printf("  Total Length: %i \n\n", ntohs(iphdr->Total_len));
 if (filelog) fprintf(file,"  Total Length: %i \n\n",
                              ntohs(iphdr->Total_len));
```

Naturalmente la manipolazione di certi campi può richiedere qualche operazione di conversione, necessarie a tradurre i valori in bit in un formato visualizzabile mediante una *printf();* nel caso degli indirizzi IP si usa ad esempio la funzione *inet_ntoa()* per ottenere delle stringhe, mentre per quanto riguarda i campi di tipo unsigned short e unsigned long ricorriamo alle rispettive funzioni *ntohs()* e *ntohl()*. Queste ultime conversioni sono necessarie per passare dal sistema di bit Big-Endian (usato nel TCP/IP) a quello Little-Endian (usato dai processori Intel).

Fig. 9: **I raw socket sono al centro di roventi polemiche.**

CATTURA E DECODIFICA DEI MESSAGGI ICMP

L'analisi dei protocolli incapsulati nell'header IP avviene in base al valore del campo iphdr->Protocol, che indica appunto il tipo di protocollo usato. La selezione avviene nel sorgente mediante un blocco switch-case, che istruisce lo sniffer su come comportarsi nell'analisi di volta in volta. Ecco ad esempio come vengono trattati i messaggi di tipo ICMP :

```
switch (iphdr->Protocol)
{ case 1:
  printf("* [Protocol: ICMP] \n");
  if (filelog) fprintf(file,"* [Protocol: ICMP] \n");
    if(filter_type==4 || filter_type==3) {
      if (Size > iphdrlen)
```

Casting

La prima cosa da fare per analizzare un pacchetto, è il casting dell'oggetto Buffer, passato come argomento dalla routine *Sniffing()*, ad un oggetto di tipo *IpHeader, che modella tutti i campi del protocollo IP.**

Sicurezza

Sicurezza
Teoria e realizzazione di uno sniffer

```
        {
            //Analyze ICMP
            icmphd = (IcmpHeader *)(Buffer + iphdrlen);
            printf("  Type: %i  ", icmphd->icmp_type);
            if (filelog) fprintf(file,"  Type: %i  ",
                                  icmphd->icmp_type);
            printf("  Code: %i  \n", icmphd->icmp_code);
            if (filelog) fprintf(file,"  Code: %i  \n",
                                  icmphd->icmp_code);
            printf("  IcmpSeq: %i \n", icmphd->icmp_seq);
            if (filelog) fprintf(file,"  IcmpSeq: %i \n",
                                  icmphd->icmp_seq);
            if(icmphd->icmp_type==0)  {printf("
                                  (EchoReply)");
              if(filelog) fprintf(file,"  (EchoReply)");
            }
              if(icmphd->icmp_type==3)  {printf
                                  ("  (DestUnreach)");
              if(filelog) fprintf(file,"  (DestUnreach)");
            }
             if(icmphd->icmp_type==8)  {printf("
                                  (EchoRequest)");
              if(filelog) fprintf(file,"  (EchoRequest)");
            }
              if(icmphd->icmp_type==11) {printf("
                                  (TimeExceed)");
              if(filelog) fprintf(file,"  (TimeExceed)");
            }
              if(icmphd->icmp_type==30) {printf("
                                  (TraceRoute)");
              if(filelog) fprintf(file,"  (TraceRoute)");}
        }
}
break;
```

Il pacchetto ICMP si trova incapsulato nel datagramma IP subito dopo la fine dell'header, pertanto bisogna inizialmente eseguire un cast a *IcmpHeader** dei dati presenti all'offset (Buffer + iphdrlen). Il resto del codice è simile a quanto visto nel protocollo IP: si accede ad ogni campo del pacchetto ICMP usando "->" e si eseguono le conversioni necessarie prima di visualizzare i dati. Per completezza si è scelto di interpretare il significato del campo *icmphd->icmp_type* (che indica il tipo di messaggio ICMP trasportato), seguendo questa tabella:

ICMP TYPE	MESSAGGIO
0	Echo Reply
3	Destination Unreachable
8	Echo Request
11	Time Exceeded
30	Trace Route

Un esempio pratico da provare è il comando PING: con lo sniffer in azione sarà possibile intercettare messaggi ICMP di tipo Echo Request ed Echo Reply scambiati fra due host.

CATTURA DEI PACCHETTI TCP E UDP

Quanto detto finora è valido, a grandi linee, anche quando il pacchetto incapsulato è di tipo TCP o UDP. Il caso UDP è abbastanza banale, perché presenta solo 4 campi; nel caso del TCP troviamo invece molte più informazioni da gestire ed è inoltre necessario qualche piccolo conticino per sapere con precisione dove inizia l'area dati, visto che nell'header TCP potrebbe esserci o non esserci il campo *Options*. Il calcolo dei limiti dell'area dati (data_start e data_end) viene fatto a partire dal valore *iphdr->Total_len* (lunghezza totale dell'intero datagramma IP) a cui bisogna sottrarre: la lunghezza dell'header IP (iphdrlen), i 20 bytes dell'header TCP e gli eventuali bytes del campo opzionale (conteggiati in data_start).

```
  //calculate data offset and length
   int data_start = ((ntohs(
                    tcphd->Off_Res_Ctrl)>>12)-5)*4;
   int data_end = ntohs(iphdr->Total_len)
                        - iphdrlen - 20 - data_start;
  printf("  Data Dump (%i): \n",data_end);
   if(filelog) fprintf(file,"  Data Dump (%i):
                                  \n",data_end);
```

Il codice utilizzato per stampare (su console e su file) i dati reali contenuti nei pacchetti (quelli dove si trovano eventuali password e codici d'accesso, per intenderci….) è un normale ciclo di for, che punta all'array di char *tcphd->dati[j]*. La stampa in formato esadecimale avviene ricorrendo a "%X" all'interno dell'istruzione printf(). Per il codice completo dell'applicazione, si rimanda al file *SPS.CPP* contenuto sul CD-ROM di ioProgrammo; la compilazione dello sniffer va eseguita con Visual C++ senza bisogno di particolari configurazioni, avendo solo l'accortezza di disporre delle classi e delle librerie Winsock2 (*winsock2.h* e *ws2_32.lib*).

```
  for(int j=data_start;j<data_end;j++)
  {
      if(!hexlog) printf("%c",tcphd->dati[j]);
      else {printf("%X ",tcphd->dati[j]);
      if(j%16==0 && j!=0) printf("\n");
  }
```

CONCLUSIONI

Si conclude qui il nostro articolo sugli sniffer e sulle tecniche di spionaggio in rete….cos'altro dire? Per una piccola prova delle potenzialità dello sniffer basta eseguirlo per una mezz'oretta su una qualsiasi rete LAN connessa tramite HUB e poi andare a dare un'occhiata ai bytes catturati, dove sicuramente appariranno pagine HTML, e-mail e password altrui. Buona "sniffata" a tutti!

Elia Florio

Esperimenti

Un esempio pratico da provare è il comando PING: con lo sniffer in azione sarà possibile intercettare messaggi ICMP di tipo Echo Request ed Echo Reply scambiati fra due host.

Biblioteca

ON LINE

DevSpy

Il sito è dedicato ai professionisti dell'IT ed è particolarmente orientato agli sviluppatori Windows. Le risorse si concentrano sul codice sorgente a tutti i livelli: articoli, script, download, progetti open-source.

http://www.devspy.com/

Developer

Il sito propone una vasta e interessante sezione di articoli e tutorial (in lingua inglese) dedicati ai più disparati ambienti di sviluppo, a partire da Java fino alla nuova tecnologia .NET

http://www.developer.com

Dev Shed

Un sito eccellente per chi è un fautore della programmazione Open Source. Una vasta serie di articoli su linguaggi e tool di sviluppo: Java, MySQL, PHP, Perl, PostgreSQL, Python,ecc.

www.devshed.com

TCP/IP – Tutto & Oltre

Un nuovo testo dedicato al protocollo di rete TCP/IP entra a far parte della ormai nota collana Tutto & Oltre di Apogeo. Il testo è una guida completa agli aspetti essenziali del protocollo TCP/IP. È rivolto a tutti coloro che vogliono conoscere in modo approfondito i dettagli che stanno dietro questo protocollo. Viene illustrato in modo dettagliato il funzionamento del protocollo, i vantaggi che esso offre e come può essere facilmente implementato. Tra gli argomenti trattati:

1 TCP/IP e Internet
2 Panoramica sui componenti della famiglia TCP/IP
3 Nomi e indirizzamenti:ARP,RARP,DNS,WINS
4 Il protocollo IPv6
5 Sicurezza della rete e del sistema
6 Configurazione di TCP/IP in ambienti Windows, Linux e Unix
7 Consigli per la risoluzione di problemi

Difficoltà: Medio – Alta • Autori: Karanjit S. Siylvan, tim Parker • Editore: Apogeo http://www.apogeonline.com • ISBN: 88-503-2044-2 • Anno di pubblicazione: 2003 Lingua: Italiano • Pagine: 862 • Prezzo: € 52,00

Professional PHP 4 Web Development Solutions

La vera potenza di PHP consiste nella sua forte propensione per lo sviluppo di applicazioni Web dinamiche. Proprio tenendo presente questo concetto, gli autori di Professional PHP 4 Web Development Solutions, hanno conferito al testo una precisa linea guida, istradando il lettore passo passo nella comprensione di tecniche che portano allo sviluppo di un sito Web dinamico: HTML, MySQL, PEAR::DB, XML e WML. È evidente l'approccio pratico adottato: ogni concetto teorico è corredato da pratici esempi in codice sorgente "pronto all'uso". Il capitolo 9 propone come realizzare un completo sistema di Content Management, mentre il capitolo 11 espone un pratico esempio su come realizzare un semplice motore di ricerca.

Difficoltà: Medio-Alta • Autore: Dash R.k. • Editore: Wrox http://www.gorilla.it ISBN: 1861007434 • Anno di pubblicazione: 2002 • Lingua: Inglese • Pagine: 613 • Prezzo: € 52

Building .NET Application for Mobile Devices

La piattaforma Microsoft .NET ha rappresentato, di fatto, una vera e propria rivoluzione per lo sviluppo di applicazioni Windows, la tecnologia, grazie al Compact Framework, ora si estende anche allo sviluppo di applicazioni per PDA e Pocket PC. Gli autori di questo testo, mostrano come utilizzare la piattaforma .NET per lo sviluppo di applicazioni sia stand-alone sia Web ,rivolte ai dispositivi mobile, mostrando un pratico utilizzo di Microsoft ASP.NET e Microsoft Visual Studio.NET, congiuntamente al Mobile Internet Toolkit. Tra gli argomenti di spicco:

- .NET per applicazioni Web Mobile
- ASP.NET e il Mobile Internet Toolkit
- Sviluppare applicazioni .NET per dispositivi mobile
- Utilizzare i Mobile Web Forms
- L'accesso ai dati
- XML Web services
- Testing e debugging utilizzando gli emulatori smartphone

Difficoltà: Media Autori: Andy Wigley, Peter Roxburgh Editore: Microsoft Press http://www.gorilla.it ISBN: 0735615322 • Anno di pubblicazione: 2002 • Lingua: Inglese • Pagine: 640 Prezzo: € 63,20 • Contiene 1 CD-Rom

Tips&Tricks

I trucchi del mestiere

La rubrica raccoglie trucchi e piccoli pezzi di codice che solitamente non trovano posto nei manuali, ma sono frutto dell'esperienza di chi programma. Alcuni trucchi sono proposti dalla Redazione, altri provengono da una ricerca sulla Rete delle Reti, altri ancora ci giungono dai lettori. Chi vuole contribuire potrà inviarci i suoi tips&tricks preferiti che, una volta scelti, verranno pubblicati nella rubrica. Il codice completo dei tips lo trovate nel CD allegato nella directory \tips\.

Visual Basic

Come posizionare il mouse su un controllo presente nel form

Questa routine può essere utilizzata per posizionare il puntatore del mouse su un particolare controllo presente all'interno di un form. Per esempio in una finestra di dialogo si può posizionare, automaticamente, il cursore sul bottone OK.
Per realizzare questa particolare funzione la routine ricorre a due API di sistema: *ClientToScreen* e *SetCursorPos*.

Come downloadare e visualizzare un'immagine dal Web

Poche righe di codice per realizzare una simpatica funzione che permette di downloadare dal Web una qualunque immagine e di visualizzare questa all'interno della propria applicazione.
Per utilizzare la funzione è necessario aggiungere al form VB il controllo INET.

Come eseguire un'applicazione nella shell e attenderne la fine dell'esecuzione

Molte volte capita di dover eseguire un programma esterno, ponendo in "pausa" la propria applicazione e attendendo la fine dell'esecuzione dell'applicazione esterna.
Il tip proposto consente proprio di realizzare questa utilissima funzione.

Come cambiare lo sfondo del desktop

E' possibile cambiare lo sfondo del desktop da un'applicazione Visual Basic? La risposta e si e di seguito ne proponiamo l'implementazione.

Come spostare un form da ogni lato

Notoriamente, cliccando su uno dei 4 lati di un generico form, si realizza l'operazione di resize del form stesso.
Grazie al tip proposto è possibile trasformazione l'operazione di resize in un'operazione di spostamento del foglio.
Sarà così possibile muovere a proprio piacimento il form semplicemente cliccando e trascinando uno qualsiasi dei suoi lati.

Delphi

Come mostrare il contenuto di una directory e delle relative sottodirectory

La procedura mostra tutti i file e le directory (incluse le sottodirectory) contenute nella directory inserita come parametro della funzione.
I risultati sono archiviati una lista di stringhe denominata List

Come ricercare del testo all'interno di un controllo Memo

Implementare una procedura per ricercare del testo all'interno di un controllo Memo (controllo all'interno del quale è possibile inserire più righe di testo), non è complicato come potrebbe apparire.
L'esempio proposto prevede che sul form sia presente un controllo *Memo* (*Memo1*) e un componente *FindDialog* (denominato *FindDialog1*).

Come ottenere la percentuale d'utilizzo del processore

Quanto sarà "indaffarato" il processore del nostro personal computer? Di seguito un utile tip che mostra, in percentuale, l'utilizzo del processore.
Il progetto prevedere che in un form siano presenti due oggetti *TButton*, due *TLabel* e un *TTimer*.

Come mappare l'hard-disk

In talune applicazioni può capitare di dover creare una unità disco logica che faccia riferimento al percorso di rete di un disco presente su un server remoto. Il tip proposto si occupa proprio di "mappare" un disco in modo del tutto automatico.

Come disabilitare logoff, task manager e shutdown

Questo tip funziona solo su sistemi Windows 2000/NT/XP e consente di disattivare le funzioni di chiusura del sistema, logoff dell'utente e *CTRL+ALT+CANC* (funzione per far apparire il task manager).
Il tutto è realizzato sfruttando funzioni del registro di sistema.

Come ricavare la lista delle applicazioni installate nel sistema

Quante e quali sono le applicazioni installate nel nostro sistema Windows? Semplice, basta da pannello di controllo richiamare installazione applicazioni! E se volessimo realizzare la medesima funzionalità in una nostra applicazione?
Semplice, basta utilizzare questo semplicissimo tip.

Confronto tra hostname

Una classe semplice che confronta due hostname verificandone l'appartenenza al medesimo host. Per la realizzazione di questa classe viene utilizzata la libreria *java.net*.

Analizziamo un URL

Come e ben noto, un URL altro non è che l'indirizzo di una pagina su Internet. La classe che segue, riceve in input un URL e restituisce le sue componenti.

Troviamo il server

La classe che segue prova tutte le porte locali di un host e verifica su quali di queste è attivo un server.

Occorrenza di una sottostringa in una stringa

Può essere utile trovare la prima occorrenza di una sottostringa all'interno di una stringa. Il tip scandisce i caratteri della stringa *s* alla ricerca di una sottostringa uguale a *t* e restituisce, se la trova, la posizione della prima occorrenza, in caso negativo restituisce -1.

Occorrenze di un carattere in una stringa

La classe seguente calcola quante volte un certo carattere "c" è contenuto all'interno di una data stringa "t".

ASP

Come conteggiare il numero di utenti presenti nel nostro sito

Modificando opportunamente il file *"Global.asa"*, così come mostrato dal tip che segue, sarà possibile conteggiare il numero di utenti che si trovano connessi sul proprio sito Web: una funzionalità semplice ma utilissima.

Come inviare un'email con ASP.NET

Utilizzando le nuove proprietà di ASP.NET ed in particolare il namespace *System.Web.Mail*, inviare un'email dal web diventa un gioco da ragazzi! Il tip fa uso della funzione *MailMessage()*.

Come reperire informazioni su un file Macromedia Flash

La classe di seguito presentata, consente di leggere alcune proprietà di un file Macromedia Flash; nel particolare le informazioni che vengono reperite sono *height*, *width*, *version*, *file length*, *twips readings*, *frame rate*, e *frame count*.

Come importare il contenuto di un file di testo in array

Lo script, facendo uso del File System Object, legge uno specifico file testuale e "immagazzina" ogni singola linea di testo in esso contenuta in una data struttura array.

Come disabilitare il tasto "Indietro" di Internet Explorer

Il tip consente di disabilitare la pressione del bottone *"Indietro"* di Internet Explorer; in particolare mostra un esempio che disabilita la pressione del tasto qualora ci si sposti da una pagina web (*page2.asp*, preventivamente invocata dalla pagina *default.asp*), alla pagina precedente.

Esperimenti DI ELETTRONICA DIGITALE: PORTE LOGICHE E OSCILLATORI

Elettronica e Delphi

Vediamo come sia possibile, al costo di un euro, realizzare un circuito elettronico che sia in grado di interfacciarsi con la porta seriale del PC e di simulare un semplice sistema di controllo: in questo modo possiamo avvicinarci alla progettazione di un oscillatore, connesso ad un circuito digitale dotato di porte logiche.

File sul CD
\soft\codice\
SpuntoLogicTester2.zip

Supponiamo di volere realizzare, a scopo didattico un circuito logico, che sia in grado di simulare un semplice circuito di controllo degli accessi. Conviene prima analizzare le specifiche del sistema richiesto, per decidere quindi la via da intraprendere. Supponiamo di dovere progettare un sistema che si basi su due livelli di controllo, ad esempio per proteggere l'ingresso di un qualunque edificio che abbia due porte di accesso in sequenza: vogliamo che all'apertura della prima venga accesa una luce all'interno dell'ingresso, mentre all'esterno dell'edificio si accende una luce lampeggiante. Desideriamo inoltre che si attivi una sirena ad intermittenza se all'apertura della seconda porta non sia stato precedentemente disabilitato il sistema per proteggere, ad esempio, l'operatore di una apparecchiatura pericolosa in funzionamento. Se l'allarme viene disabilitato, vogliamo che la sirena e tutte le luci vengano disattivate. Vista la relativa complessità del sistema, si potrebbe pensare che si renda necessario un sistema a microprocessore; vediamo invece come il problema sia risolvibile, con un comunissimo circuito integrato dotato di quattro porte logiche e con pochi altri componenti discreti: il tutto ad un costo di circa un euro.

LE PORTE LOGICHE NAND: L'INTEGRATO 4011

Il lettore di queste pagine, sicuramente programmatore esperto, è senz'altro familiare con il costrutto logico 'IF THEN ELSE', nonché con i vari operatori logici 'AND, OR e NOT'; ebbene esiste una famiglia di circuiti integrati che permette di eseguire operazioni logiche autonomamente, senza l'ausilio del microprocessore. Volendo fare un esempio pratico, supponiamo di avere due pulsanti, che chiamiamo A e B, ed una lampadina che identifichiamo con Y: inoltre, per nostra convenzione, stabiliamo che la pressione del pulsante e l'accensione della lampadina corrisponda ad un livello logico '1', mentre il contrario ad un livello logico '0'. In Fig. 1 si riportano le connessioni dell'integrato 4011, contenente al proprio interno quattro porte logiche NAND a due ingressi (cortesia Philips Semiconductors), gli ingressi sono chiamati I1-I8, mentre le uscite O1-O4; è da notare che l'operazione logica effettuata da una porta non influisce as-

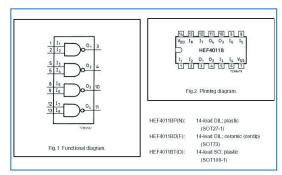



Fig. 1: Schema logico e la piedinatura del circuito integrato HEF4011, reperibili in forma completa su Internet all'indirizzo: http://www.components .philips.com/ (cortesia Philips Semiconductors).

Le Porte logiche NAND

Le porte logiche NAND hanno la caratteristica di presentare in uscita un livello logico corrispondente alla operazione logica NOT AND delle linee di ingresso. Per una porta NAND a due ingressi, chiamati ad esempio A e B, all'uscita Y troveremo un livello logico corrispondente a : Y=NOT(A AND B).

A	B	Y= NOT A	Y= NOT B	Y=A OR B	Y=A AND B	Y=A NOR B	Y=A NAND B
0	0	1	1	0	0	1	1
0	1	1	0	1	0	0	1
1	0	0	1	1	0	0	1
1	1	0	0	1	1	0	0

Tab. 1: La *"Tabelle della verità"* si ha un riassunto dei risultati logici delle operazioni logiche NOT, OR ed AND, in aggiunta a queste, molto utilizzate in elettronica, troviamo anche NOR (NOT OR) e NAND (NOT AND), che si riferiscono semplicemente alla negazione logica delle operazioni OR ed AND.

Fig. 2: Per proteggere i circuiti integrati CMOS, è opportuno conservarli sugli appositi supporti antistatici, in figura si riporta l'integrato HEF4011 utilizzato nell'articolo (cortesia Philips Semiconductors).

solutamente con quelle delle altre. Dicevamo che le porte NAND e NOR sono molto più diffuse delle porte AND ed OR e ci si potrebbe chiedere perché, dal momento che le seconde sono, in effetti le operazioni logiche fondamentali. Il motivo è essenzialmente pratico: osservando il simbolo della porta logica, si nota un pallino (il simbolo della negazione logica) posto in prossimità del terminale di uscita: significa che l'operazione logica che avviene all'interno della porta è Y=NOT(A AND B), quindi se colleghiamo elettricamente A e B insieme (ponendo quindi A=B) la relazione precedente diventa Y=NOT(A AND A) , ovvero Y=NOT A: abbiamo costruito una porta logica NOT, collegando gli ingressi di una porta NAND. La maggiore diffusione di porte NAND si spiega con la loro maggiore flessibilità, dal momento che permettono di ottenere, con un solo microchip dotato di quattro porte tutte le configurazioni logiche.

LO SCHEMA ELETTRICO

Analizzando lo schema elettrico della nostra realizzazione, possiamo notare innanzi tutto che viene collegato alla porta seriale del PC: i segnali relativi sono riportati di seguito, corredati di abbinamento tra il segnale della porta seriale ed il significato nella nostra applicazione: per comodità nello schema elettrico sono espresse tra parentesi le connessioni per la porta universale già proposta in questa rivista (Tab. 2).
Per chi si chiede come venga alimentato il circuito, possiamo dire che viene prelevata corrente elettrica da una linea di uscita della porta (DTR), lo stesso metodo utilizzato nei mouse seriali: l'alimentazione fluisce attraverso il diodo D1, che lascia passare corrente quando DTR è a livello logico 1 (+10 V in genere) ma la blocca quando è a livello logico '0' (-10 V): lo stesso concetto vale anche per la linea RTS, questa volta utilizzata come segnale logico da inviare alla logica della nostra porta. La massa logica della porta seriale viene collegata alla massa del nostro circuito. Ricordiamo che gli integrati CMOS possono essere alimentati generalmente con una tensione compresa tra +5 e +15 V, è consigliabile consultare il Data Sheet del costruttore per ulteriori dettagli.
Nella parte alta dello schema, notiamo le due porte

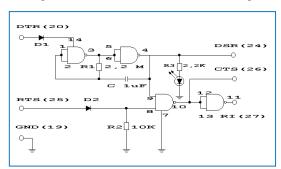



Fig. 3: Lo schema elettrico riportato in figura può essere suddiviso concettualmente in due parti: un oscillatore a bassa frequenza, costituito dalle due porte logiche in alto ed un circuito logico, comprendente le due porte in basso.

logiche collegate come oscillatore, o multivibratore astabile: questa configurazione permette di generare una forma d'onda rettangolare: la frequenza di uscita viene determinata con la formula: $f=0,72/(R*C)$, nel nostro caso corrispondente a circa 0,5 Hertz, valore idoneo al lampeggiamento delle luci esterne del nostro avvisatore ottico e per la nostra sirena. All'uscita all'oscillatore troviamo un diodo LED, che simula il lampeggiante posto all'esterno della nostra infrastruttura, mentre le due porte logiche nella parte bassa del circuito si occupano della gestione delle uscite relative alla luce dell'ingresso (CTS) ed all'attivazione della sirena (RI). I numeri riportati tra parentesi rappresentano i terminali della scheda universale proposta su ioProgrammo del mese di Novembre 2002.

Porta Seriale: Connettore 25 PIN	Porta Seriale: Connettore 9 PIN	Segnale PortaSeriale (Simulazione del Circuitodi Controllo)	Tipo di segnale Porta seriale
Pin 4	Pin 7	RTS (Apertura seconda porta)	Request To Send
Pin 5	Pin 8	CTS (Luce Ingresso)	Clear To Send
Pin 6	Pin 6	DSR (Luce lampeggiante esterna)	Data Set Ready
Pin 7	Pin 5	SG (GND, massa elettrica)	Signal Ground
Pin 20	Pin 4	DTR (Attivazione sistema di controllo)	Data Terminal Ready
Pin 22	Pin 9	RI (Sirena)	Ring Indicator

Tab. 2: La *"Tabelle della verità"* si ha un riassunto dei risultati logici delle operazioni logiche NOT, OR ed AND, in aggiunta a queste, molto utilizzate in elettronica, troviamo anche NOR (NOT OR) e NAND (NOT AND), che si riferiscono semplicemente alla negazione logica delle operazioni OR ed AND.

Piastre per montaggi sperimentali

Per la realizzazione di questo circuito è stata utilizzata la 'scheda universale' proposta su questa rivista nel numero di Novembre 2002, sulla quale si possono trovare informazioni all'indirizzo: http://web.tiscali.it/spuntosoft/. E' possibile comunque utilizzare una comune breadboard per montaggi sperimentali.

Elettronica e Delphi

Esperimenti di Elettronica Digitale: Porte Logiche e Oscillatori

I componenti necessari:

N 1 CMOS 4011
D1, D2 N2 Diodi 1N4148
R1 N1 Res. 2,2 MOhm
R2 N1 Res. 10 KOhm
R3 N1 Res. 2,2 KOhm
C N1 Condensatore 1 uF al Tantalio.

Precauzioni

Prima di collegare il circuito al nostro PC occorre verificare la nostra realizzazione con attenzione per assicurarci che tutto sia stato collegato come previsto.

L'oscillatore

L'oscillatore presentato in questa sede è un generatore di onda rettangolare, semplice ed economico che può essere utilizzato anche per altri scopi.

REALIZZAZIONE DELL'OSCILLATORE

Per procedere alla realizzazione del circuito è possibile utilizzare la scheda universale già proposta su questa rivista. In alternativa è possibile procedere al montaggio del circuito utilizzando una comune scheda per montaggi sperimentali chiamata Breadboard, reperibile nei negozi di componenti elettronici.

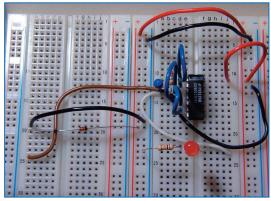



Fig. 4: La vista di insieme dell'oscillatore comprende il circuito integrato, la sezione di alimentazione e la parte optoelettronica dotata di diodo LED.

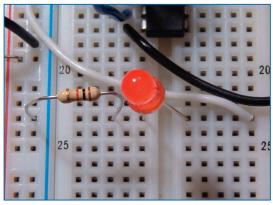
In Fig. 4 è riportata la vista di insieme del circuito oscillatore, come si può notare, la realizzazione del circuito avviene senza saldature, dal momento che tutti i collegamenti vengono eseguiti con spezzoni di filo elettrico rigido. Nella parte destra si può notare il circuito integrato e la componentistica relativa all'oscillatore, mentre nella parte sinistra troviamo i pochi componenti della sezione di alimentazione. In un secondo momento verranno installate le connessioni relative alla logica del circuito. In prossimità del circuito integrato troviamo le connessioni relative all'oscillatore; notiamo in particolare i collegamenti dell'alimentazione (fili rossi e neri), nonché della parte relativa alle porte logiche, realizzate con fili elettrici di colore blu. Nella sezione relativa all'alimentazione, notiamo il diodo che ha il compito di prelevare il segnale relativo a DTR, che viene utilizzato per alimentare il circuito. La corrente prelevabile da una porta seriale varia a seconda del tipo di elettronica che è installata nel computer, che è funzione del tipo di computer stesso: è probabile che per un portatile, ad esempio, la corrente prelevabile possa essere sensibilmente inferiore rispetto ad un desktop. Per correnti fino a pochi milliamperes, ed a meno di casi molto particolari, non dovremmo avere problemi. Il diodo LED ha lo scopo di visualizzare, accendendosi, la presenza di un impulso positivo all'uscita dell'oscillatore; è possibile anche verificare il funzionamento del resto del circuito, scollegando il filo giallo mostrato in figura e riconnettendolo alle uscite delle altre porte logiche, per controllarne il funzionamento. E' importante controllare che sia inserita la resistenza da 2,2 KOhm a valle, oppure a monte del LED, allo scopo di limitare la corrente all'interno del diodo luminoso: è altresì di fondamentale importanza controllare che il LED sia collegato con la giusta polarità, ovvero con il terminale più corto collegato dal lato della massa elettrica del circuito.

Dopo avere realizzato tutto il circuito, prima della connessione al PC verifichiamo che tutte le connessioni siano state effettuate correttamente, in quanto un cablaggio errato può portare in casi estremi al danneggiamento della porta seriale, o peggio della scheda madre del Computer. In considerazione di quanto detto si raccomanda al lettore la massima cura nella realizzazione del circuito, con gli accorgimenti di rito: se non siamo proprio esperti di cablaggi elettronici, facciamoci aiutare da chi è più pratico. Evitiamo assolutamente di operare modifiche o di toccare il circuito con il PC alimentato; nel caso di malfunzionamenti provvediamo a spegnere e scollegare tutto prima di lavorare sul circuito elettrico.

Fig. 5: Il diodo LED permette di visualizzare gli impulsi generati dal circuito integrato.

REALIZZAZIONE DEL CIRCUITO LOGICO

Per completare la parte logica del circuito, occorre effettuare le connessioni che vengono riportate in Fig. 6; i fili blu consentono i collegamenti tra le porte logiche, quelli gialli corrispondono ai collegamenti relativi alla linea DSR, mentre quelli rossi e bianchi, rap-

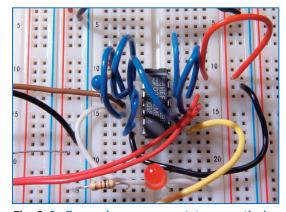



Fig. 6: In figura viene rappresentato un particolare relativo alla realizzazione del circuito logico, in prossimità del microchip.

presentano le linee di ritorno verso la porta seriale, che permettono la visualizzazione dei segnali provenienti dal circuito. Le connessioni delle linee di I/O della porta seriale permettono l'analisi dei segnali dal punto di vista logico, anche se non analogico, come nel caso di utilizzo di un comune oscilloscopio. Il vantaggio di ciò è l'elevato numero di canali analizzabili contemporaneamente. Si può notare che è stato aggiunto un secondo diodo, proveniente dalla linea DTS, che ha il compito di comandare la logica del circuito, simulando l'apertura della seconda porta della nostra simulazione.

VERIFICA DEI SEGNALI ALL'OSCILLOSCOPIO LOGICO

Una volta completata la nostra realizzazione, siamo giunti al momento di collaudarne il funzionamento, verificandone i segnali di ingresso/uscita. Provvediamo a verificare un'ultima volta tutte le connessioni elettriche, nonché le polarità dei diodi, compreso il LED; completato il controllo, siamo pronti a collegare il circuito alla porta seriale del PC, ovviamente a computer rigorosamente spento. Accendiamo il calcolatore e lanciamo il programma di monitor della porta seriale, contenuto nel file *'SpuntoLogicTester.Zip'* reperibile nel CD allegato alla rivista e provvediamo subito a selezionare la porta seriale sulla quale è collegato il circuito da verificare. Con questa operazione possiamo già verificare lo stato delle linee di ingresso/uscita della porta provvedendo, dopo avere premuto il tasto *'Enable Monitoring'*, a disabilitare DTR ed RTS per mezzo dei pulsanti appositi ed attivando la pulizia dello schermo dell'oscilloscopio con la pressione del tasto 'Clear Scope'. Le semplici operazioni che abbiamo appena compiuto ci permettono di inizializzare l'oscilloscopio per la verifica del circuito, che a questo punto risulta completamente privo di alimentazione, dal momento che le due linee di uscita sono a livello logico '0'. Premiamo ora il pulsante 'Oscilloscope Sweep' ed alimentiamo il circuito premendo il tasto DTR: con questa operazione alimentiamo l'oscillatore che provvede a generare una onda rettangolare, sulla linea DSR. In questo modo simuliamo l'apertura della prima porta e l'accensione del lampeggiante all'esterno dell'edificio. Notiamo inoltre che CTS, che rappresenta la luce dell'ingresso, nella nostra simulazione viene immediatamente posto a livello logico '1', come previsto dalle nostre specifiche. Supponiamo a questo punto che venga aperta anche la seconda porta senza che il sistema venga disabilitato, ovviamente questa situazione corrisponde a forzare DTR e RTS entrambi allo stato logico '1', in questo caso si ha l'abilitazione della sirena ad intermittenza, corrispondente all'onda rettangolare presente su RI ed il concomitante lampeggiare della luce collegata a CTS. I più attenti noteranno che l'onda rettangolare presente su CTS è la negazione logica del segnale presente in questa fase su RI, come è chiaramente visibile analizzando lo schema elettrico del circuito. Non analizziamo il comportamento del circuito nel caso in cui DTR=0 ed RTS=1, dal momento che non è contemplato dalle nostre specifiche. Nell'ultima parte del grafico, disabilitando RTS, notiamo che la sirena si spegne immediatamente, fatto poco realistico nel caso di un sistema reale di controllo, che dovrebbe prevedere almeno un sistema di ritardo che prolunghi l'allarme acustico, argomento interessante che affronteremo nel prossimo appuntamento, quando analizzeremo i sistemi Timer e di ritardo. Il collaudo dell'apparecchiatura si può considerare terminato, è possibile visualizzare lo stato logico delle varie parti del circuito, semplicemente collegando il diodo LED nelle parti che si vogliono analizzare; è inoltre possibile variare la frequenza dell'oscillatore cambiando i valori della resistenza R1 e del condensatore C, utilizzando la formula proposta in precedenza.

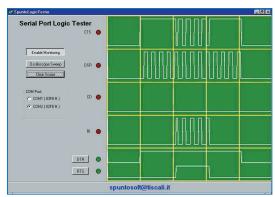



Fig. 7: L'analisi della risposta del circuito ai livelli logici presenti sulle linee DTR ed RTS è possibile correlando i segnali ricevuti sulle linee di ingresso DSR,CTS ed RI per mezzo del nostro oscilloscopio logico.

CONCLUSIONI

Abbiamo visto, in queste pagine, come sia possibile realizzare un circuito logico dotato di un oscillatore in grado di generare un'onda rettangolare della quale possiamo variare la frequenza di uscita. È stato simulato, inoltre, un semplice circuito di controllo, applicando la teoria dell'algebra booleana ad un circuito elettronico 'solido', in grado di funzionare anche senza l'ausilio del PC.

Il lettore vorrà comprendere che nonostante quanto esposto in queste pagine sia stato debitamente verificato e collaudato, tuttavia viene riportato a scopo illustrativo e di studio, pertanto l'editore e l'autore non sono da considerare responsabili per eventuali conseguenze derivanti dell'utilizzo di quanto esposto in questa sede, soprattutto per la tipologia e la complessità dell'argomento. L'autore è lieto di rispondere a qualunque domanda e ad accettare qualunque tipo di consiglio.

Luca Spuntoni
(spuntosoft@tiscalinet.it)

Elettronica e Delphi

Esperimenti *di Elettronica Digitale: Porte Logiche e Oscillatori*

L'Oscilloscopio

L'Oscilloscopio è uno strumento indispensabile per lo sperimentatore elettronico, il programma utilizzato in queste pagine visualizza soltanto segnali digitali, caratteristica eccellente per l'analisi del circuito proposto. Il programma è contenuto nel file *'SpuntoLogicTester.Zip'*, ed è utilizzabile semplicemente estraendolo in una apposita directory e lanciando il programma eseguibile *'SpuntoLogic TesterProject.exe'*.

Sul Web

Philips Components:
http://www.components.philips.com/

Informazioni sulla scheda universale:

http://web.tiscali.it/spuntosoft/

Telecontrollo VIA WEB

Micro Controller

Grazie ai moduli Rabbit è possibile realizzare applicazioni di telecontrollo Web in pochi e semplici passi. Comandare ogni dispositivo elettrico direttamente da una pagina Internet diventa un gioco da ragazzi...

File sul Web
www.itportal.it/Rabbit/Led_Web.zip.

In uno dei precedenti articoli, abbiamo esaminato il funzionamento dei moduli Rabbit, veri e propri gioiellini elettronici in grado di interfacciare il PC con qualunque dispositivo elettrico. Un completo sistema a microprocessore (dalle dimensioni ridottissime) in grado di "ospitare" un server Web e offrire finanche il telecontrollo di dispositivi da una normale pagina Web.
Pensate per esempio alla possibilità di accendere o spegnere l'impianto di condizionamento della propria abitazione, visitando una pagina Internet da un qualunque PC del mondo connesso in Rete. Se nel precedente articolo (apparso nel n° 66 di ioProgrammo), ci siamo soffermati nello spiegare le caratteristiche dei moduli Rabbit, in questo secondo appuntamento, dando voce alle molte richieste dei nostri lettori, è nostro compito soffermarci sull'implementazione pratica di un primo progetto che sfrutti le potenzialità del modulo elettronico.
Il progetto che andremo ad esaminare mostra come controllare lo stato di una linea digitale attraverso una normale pagina Web.

Fig. 1: Modulo Rabbit.

Distribuito da...
Area SX S.r.l.
INFORMATICA & MICROELETTRONICA
http://www.areasx.com
info@areasx.com
Via Luigi Robecchi Brichetti, 13
00154 - Roma
Tel. 06 57.17.26.79
Fax. 06 57.17.26.95

CONTROLLO DI UN LED "REMOTO" DA UNA PAGINA WEB

Il progetto che andremo a realizzare mostra come controllare da una pagina Web lo stato (acceso /spento) dei due Led posti sulla scheda SX01 (la scheda fornita a corredo del modulo Rabbit per realizzare i propri esperimenti elettronici – vedi ioProgrammo n°66), si tratta di un semplice progetto che potrà funzionare da base per la realizzazione di un qualcosa di più concreto come il controllo dell'impianto di luce, la caldaia di riscaldamento, l'attivazione dell'impianto elettronico dell'antifurto.

COSA OCCORRE PER IL PROGETTO

L'hardware che andremo ad utilizzare sarà: un modulo Rabbit RCM2200, una scheda SX01 ed un normale cavo di Rete; nel caso in cui si dispone di una rete locale (LAN) si potrà utilizzare per il collegamento un cavo di Rete normale senza la necessità di impiegare un cavo incrociato (cross-cable). Il modulo Rabbit RCM2200 è dotato di linee di I/O per pilotare i diodi led, di uno stack TCP/IP per ospitare una pagina Web ("immagazzinati" in una comune memoria di tipo flash), fungendo quindi da server Web.

IL CODICE C DEL PROGETTO

Di seguito viene proposto e successivamente discusso, il codice sorgente da scaricare nel modulo Rabbit RCM2200; il codice consentirà di implementare la funzione di accensione/spegnimento dei led direttamente da una pagina Web. Affinché il codice possa essere interpretato e "scaricato" nel modulo Rabbit, è necessario utilizzare l'apposito ambiente di programmazione Dynamic C:

```
/*********************************************
 Autore: Daniele De Santis e-mail: desantis@areasx.com
*********************************************/
//Impostazioni del TCP/IP
#define MY_IP_ADDRESS   "10.1.1.2."
#define MY_NETMASK      "255.255.255.248"
//impostazioni del server WEB
#define TCP_BUF_SIZE 2048
#define HTTP_MAXSERVERS 1
#define MAX_TCP_SOCKET_BUFFERS 1
```

```
#define REDIRECTHOST MY_IP_ADDRESS
#define SSPEC_MAXSPEC 19
#define HTTP_NO_FLASHSPEC
#define REDIRECTTO "http://" REDIRECTHOST
                                          "/index.shtml"
//Carica le librerie
#memmap xmem
#use "dcrtcp.lib"
#use "http.lib"
//importa i file nella memoria del RABBIT
#ximport "C:/RABBIT/LED_WEB/pages/web.shtml" index_html
#ximport "C:/RABBIT/LED_WEB/pages/ledon.gif" ledon_gif
#ximport "C:/RABBIT/LED_WEB/pages/ledoff.gif" ledoff_gif
#ximport "C:/RABBIT/LED_WEB/pages/button.gif" button_gif
#ximport "C:/RABBIT/LED_WEB/pages
          /logo_rabbit_small.gif" logo_rabbit_small_gif
//Setta i tipi di file usati
const HttpType http_types[] = {
  { ".shtml", "text/html", shtml_handler}, // ssi
  { ".html", "text/html", NULL},           // html
  { ".cgi", "", NULL},                     // cgi
  { ".gif", "image/gif", NULL} };
char led1[15];
char led2[15];
int led1toggle(HttpState* state){
  if (strcmp(led1,"ledon.gif")==0) {
    strcpy(led1,"ledoff.gif");
    BitWrPortI(PDDR, &PDDRShadow, 0, 3); } else {
    strcpy(led1,"ledon.gif");
    BitWrPortI(PDDR, &PDDRShadow, 1, 3); }
  cgi_redirectto(state,REDIRECTTO);
  return 0;  }
//Gestione del LED LED TEST 1
int led2toggle(HttpState* state) {
  if (strcmp(led2,"ledon.gif")==0) {
    strcpy(led2,"ledoff.gif");
    BitWrPortI(PBDR, &PBDRShadow, 0, 7); } else {
    strcpy(led2,"ledon.gif");
    BitWrPortI(PBDR, &PBDRShadow, 1, 7); }
  cgi_redirectto(state,REDIRECTTO);
  return 0; }
main() {
  //carica i file in memoria
  sspec_addxmemfile("/", index_html, SERVER_HTTP);
  sspec_addxmemfile("index.shtml", index_html,
                                        SERVER_HTTP);
  sspec_addxmemfile("ledon.gif", ledon_gif,
                                        SERVER_HTTP);
  sspec_addxmemfile("ledoff.gif", ledoff_gif,
                                        SERVER_HTTP);
  sspec_addxmemfile("button.gif", button_gif,
                                        SERVER_HTTP);
  sspec_addxmemfile("logo_rabbit_small.gif",
               logo_rabbit_small_gif, SERVER_HTTP);
  sspec_addvariable("led1", led1, PTR16, "%s",
                                        SERVER_HTTP);
  sspec_addvariable("led2", led2, PTR16, "%s",
                                        SERVER_HTTP);
  sspec_addfunction("led1tog.cgi", led1toggle,
                                        SERVER_HTTP);
  sspec_addfunction("led2tog.cgi", led2toggle,
                                        SERVER_HTTP);
  WrPortI(PDFR, &PDFRShadow, 0x0);
  WrPortI(PDDCR, &PDDCRShadow, 0x0);
  WrPortI(PDDDR, &PDDDRShadow, 0xff);
        strcpy(led1,"ledoff.gif");
        strcpy(led2,"ledoff.gif");
        sock_init();
        http_init();
        tcp_reserveport(80);
        while (1) { http_handler();}
}
#nodebug
```

Il sorgente proposto nelle prime righe contiene una serie di parametri di configurazione, ognuno di questi dovrà essere personalizzato a seconda delle proprie esigenze. Un parametro che necessita di particolare attenzione è quello dedicato alla gestione dell'indirizzo IP: (*MY_IP_ADDRESS* e *MY_NETMASK*), è proprio attraverso tali parametri che il modulo Rabbit si adatterà alla Rete in cui lo stesso dispositivo sarà collegato. Nel caso di collegamento con cavo incrociato, ad un singolo PC, è possibile scegliere due indirizzi IP generici (ad es. 10.1.1.1 per il PC e 10.1.1.2 per il Rabbit). Nel caso invece in cui si vorrà collegare l'apparato ad una LAN esistente, occorrerà conformarsi alla classe di indirizzi in uso sulla LAN stessa. I parametri che controllano queste configurazioni sono:

```
#define MY_IP_ADDRESS "10.1.1.2"
#define MY_NETMASK "255.255.255.248"
```

Per il nostro scopo gli altri parametri di configurazione possono essere lasciati invariati. Le righe del tipo:

```
#ximport "C:/RABBIT/LED_WEB/pages/web.shtml"
                                        index_html
```

specificano la posizione e i file da uploadare all'interno della memoria flash del modulo Rabbit; nel nostro caso abbiamo indicato la pagina Web *web.shtml* e tre immagini GIF di supporto. All'interno del sorgente *LED_WEB.C*, sono presenti alcune voci che specificano il percorso dei file che devono essere caricati nella memoria del Rabbit; tali percorsi vanno modificati in base alla posizione dei propri file sull'hard-disk:

```
#ximport "C:/LED_CONTROL/pages/web.shtml" index_html
#ximport "C:/LED_CONTROL/pages/ledon.gif" ledon_gif
...
```

La pagina *web.shtml* è una classica pagina Web scritta quasi interamente in linguaggio HTML standard, di seguito è mostrato il codice per intiero:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD W3
```

Micro Controller

Telecontrollo
via Web

Controllo LED Remoto

Il progetto che andremo a realizzare mostra come controllare da una pagina Web lo stato (acceso/spento) dei due Led posti sulla scheda SX01

Indirizzo IP

Un parametro che necessita di particolare attenzione è quello dedicato alla gestione dell'indirizzo IP: (MY_IP_ADDRESS e MY_NETMASK), è proprio attraverso tali parametri che il modulo Rabbit si adatterà alla Rete in cui lo stesso dispositivo sarà collegato.

Micro Controller

Telecontrollo *via Web*

```
HTML//EN">
<HTML>
<HEAD>
  <TITLE>Esempio di controllo LED dal WEB</TITLE>
  <meta http-equiv=refresh content="10 ;
                                   url=/index.shtml">
</HEAD>
<BODY topmargin="0" leftmargin="0"
              marginwidth="0" marginheight="0"
  bgcolor="#FFFFFF" link="#009966" vlink=
                        "#FFCC00" alink="#006666">
<CENTER>
<H1><font face="Arial">WEB CONTROL</font></H1>
<HR>
<TABLE BORDER="0" CELLSPACING="2"
                              CELLPADDING="1">
<TR>
  <TD ALIGN="CENTER"> <img SRC="<!--#echo var=
          "led1"-->" ALT="LED TEST 1"> </TD>
  <TD ALIGN="CENTER"> <img SRC="<!--#echo
        var="led2"-->" ALT="LED TEST 2"> </TD>
</TR>
<TR>
  <TD ALIGN="CENTER"> <A HREF="/led1tog.cgi">
                 <img SRC="button.gif" BORDER=0>
</A> </TD>
   <TD ALIGN="CENTER"> <A HREF="/led2tog.cgi">
                 <img SRC="button.gif" BORDER=0>
</A> </TD>
</TR>
</TABLE>
<HR>
</CENTER>
</BODY>
</HTML>
```

L’intero progetto, è scaricabile dall’indirizzo web: *www.itportal.it/Rabbit/Led_Web.zip.*

COME COMPILARE L’APPLICAZIONE

Per poter realizzare il progetto è necessario memorizzare il programma *LED_WEB.C* all’interno del modulo Rabbiit, questo, come già accennato in precedenza, è possibile grazie all'ambiente di sviluppo fornito da Rabbit Semiconductor nel suo Development Kit. Una volta caricato il sorgente nell'ambiente di sviluppo e portata a termine con successo la compilazione, il Web server contenuto nel modulo si pone in attesa di connessioni, all'indirizzo IP assegnato in precedenza.

Per compilare l’applicazione si consiglia, preventivamente, di azzerare la memoria del Rabbit, caricando il BIOS di default; tale operazione la si realizza, collegando opportunamente il modulo Rabbit al PC (consultare l’articolo apparso sul numero 66 di ioProgrammo) e selezionando dal menu *Compile*, di Dynamic C, la voce *Reset Target/Compile* BIOS o più semplicemente mediante la pressione del tasti *CTRL+Y*. Successivamente si potrà procedere nell’uploadare il sorgente, ed i relativi file di supporto, all’interno della memoria flash del Rabbit. Per avviare la compilazione, sarà necessario selezionare, sempre dal menu *Compile*, la prima voce in elenco, ovvero *Compile to Target*, o premere il tasto funzione F5.

Compilazione

Per compilare l’applicazione si consiglia, preventivamente, di azzerare la memoria del Rabbit, caricando il BIOS di default; tale operazione la si realizza, collegando opportunamente il modulo Rabbit al PC.

Fig. 2: La compilazione del file led_web.c

Effettuata la compilazione, generalmente bastano una diecina di secondi, per verificare che tutto funzioni sarà sufficiente avviare il browser e digitare l'indirizzo IP assegnato al modulo. Nel nostro caso digiteremo *http://10.1.1.2* se tutto è andato a buon fine vedremo comparire la pagina mostrata in Fig. 3.

Fig. 3: La pagina Web “caricata” nel modulo Rabbit.

Cliccando su uno dei due pulsanti presenti nella pagina Web e posti in corrispondenza dell’immagine dei due Led (Verde e Rosso) noteremo contestualmente che uno dei due led collocati sulla scheda SX01 si accenderà o spegnerà. In questo primo progetto ci siamo limitati a dimostrare come il modulo Rabbit possa essere impiegato per creare delle pagine Web in grado di controllare dei semplici diodi Led, nei prossimi appuntamenti amplieremo il discorso mostrando come controllare dei veri e propri apparati elettrici, e come allargare il controllo non solo ad una semplice pagina Web bensì a vere e proprie applicazioni Visual Basic; finiremo quindi per esaminare come offrire ancor più integrazione tra l’utente ed il modulo Rabbit grazie all’interfacciamento con Macromedia Flash. Preparatevi a controllare il mondo.

Excel e VBA

SOLUZIONI ED ESEMPI DI CODICE

Sistema

Imparare "per esempi" a programmare Excel con Visual Basic for Applications

Office offre ai propri utenti enormi potenzialità. Inoltre dispone di un linguaggio di programmazione intuitivo e potente: VBA (Visual Basic for Applications). In questo articolo presenteremo dei semplici programmi o macro per risolvere alcuni problemi che si possono presentare nell'uso di Excel. L'obiettivo è dare spunto per risolvere altri problemi simili, riutilizzando i frammenti di codice presentati.

File sul CD

Sul CD allegato alla rivista potete trovare il documento vba.xls contenente dei dati di prova e le macro descritte nell'articolo.

MACRO

Spesso, usando Office (in particolare presenteremo il caso di Excel), si ha la necessità di eseguire operazioni ripetitive sui valori memorizzati. Per esempio si può avere la necessità di estrapolare certe informazioni che sono memorizzate insieme ad altre (si pensi al caso di colonne che memorizzano una descrizione insieme ad un codice, e che si voglia prendere il codice e memorizzarlo in una colonna separata), oppure raggruppare informazioni sparse su più colonne, magari elaborandole. Una delle possibili soluzioni per risolvere questo tipo di problemi è quella di creare delle macro. Una macro è un frammento di codice che viene mandato in esecuzione al verificarsi di un particolare evento (di solito la pressione di una combinazione di tasti). Il modo più semplice di realizzare una macro è quello di usare il "Registratore di macro" (menu *Strumenti > Macro > Registra nuova macro*, Fig. 1). Per registrarla è necessario dare un nome alla macro e associare ad essa una pressione di tasti che la

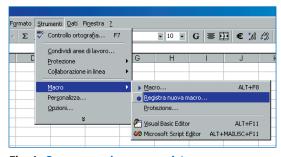



Fig. 1: **Come procedere per registrare una nuova macro.**

mandi in esecuzione (*Ctrl+"tasto"*). Supponendo di voler eliminare tutte le righe del foglio di lavoro che contengono la dicitura *"[cancella]"*, selezioniamo *"Registra nuova macro"*, chiamiamo la nuova macro *"eliminaRighe"* e vi associamo la combinazione *Ctrl+r*. Eseguiamo poi i seguenti passi:

1) Dal menu *"Modifica > Trova"*, cerchimo la parola *"[cancella]"* e premiamo *"Trova successivo"*.
2) Chiudiamo la finestra di ricerca (tasto *"Chiudi")*.
3) Dal menu *"Modifica"* selezioniamo *"Elimina"* e poi selezioniamo *"Sposta le celle in alto"*.
4) Premiamo sul tasto di "Interrompi registrazione" sulla finestra della macro (Fig. 2).

Fig. 2: **Tasto per interrompere la registrazione della macro.**

A questo punto, premendo ripetutamente *"Ctrl+r"*, verranno via via cancellate tutte le righe che contengono la parola *"[cancella]"*. Questo sistema permette di risolvere molti piccoli problemi, è semplice anche per chi non conosce la programmazione e, dulcis in fundo, può essere un ottimo metodo per imparare a programmare in VBA "per esempi". Infatti se andate sul menu *"Strumenti > macro > macro"* (o premete *Alt+F8*), vi apppaiono tutte le macro presenti. Selezionate la macro che abbiamo appena creato e premete il tasto *"Modifica":* appare l'editor di Visual Basic con il codice sorgente della macro (*Listato 1*). Potete "divertirvi" a creare alcune macro con il "Registratore di Macro" e poi andare a curiosare il codice generato.

Listato 1: La macro generata dal "Registratore di Macro".

```
Sub cancellaRiga()
'
' cancellaRiga Macro
' Macro registrata il 01/02/2002 da Standard
'
' Scelta rapida da tastiera: CTRL+r
'
    Cells.Find(What:="[cancella]", After:=ActiveCell, _
               LookIn:=xlFormulas, LookAt:=xlPart, _
```

```
    SearchOrder:=xlByRows, SearchDirection:=xlNext, _
                                  MatchCase:=False).Activate
   Cells.FindNext(After:=ActiveCell).Activate
   Selection.Delete Shift:=xlUp
End Sub
```

INIZIAMO A PROGRAMMARE

Per problemi più complessi di quelli visti in precedenza non basta registrare le macro nel modo che abbiamo illustrato, ma è necessario ricorrere alla loro programmazione. Chi si avvicina alla programmazione di VBA, si trova ad affrontare due difficoltà: la prima è quella di imparare il linguaggio, la sua sintassi e i suoi costrutti fondamentali; la seconda è capire quali strumenti e funzionalità mette a disposzione l'applicazione che si vuole personalizzare (Excel, Word, o qualsivoglia altro programma presente in Office). Per risolvere entrambi i problemi è fondamentale avere a disposizione della buona documentazione. Chi ha ottime basi di programmazione troverà semplice risolvere il primo poblema, mentre per risolvere il secondo basta avere esperienza come utenti dell'applicazione (quando si conosce il funzionamento di certe operazioni risulta più semplice imparare ad usarle). In questo articolo daremo per scontato la conoscenza dei fondamenti della programmazione e del Basic (in particolare del Visual Basic), mentre approfondiremo l'uso di alcune delle (molte) funzionalità esposte da Excel. Chi volesse ulteriori risorse può far riferimento alla bibliografia.

USARE LE FUNZIONALITÀ ESPOSTE DA EXCEL

Ogni componente di Office espone numerosi oggetti, proprietà, metodi, eventi e insiemi di oggetti utili a creare applicazioni personalizzate. Gli oggetti sono organizzati in una struttura gerarchica. Gli oggetti esposti da una applicazione sono utilizzabili anche nelle altre. La tecnologia di base che permette tutto ciò è COM. Il programmatore può far uso di tali funzionalità attraverso VBA o altri linguaggi di programmazione. Office ha al suo interno un ambiente di programmazione VBA, che rende agevole la scrittura di programmi in tale linguaggio.
Per usare le proprietà di un oggetto è necessario aver specificato un riferimento ad esso.
Vediamo quali sono gli oggetti principali.

- **Oggetto Application:** è l'oggetto principale di tutte le applicazioni Office. La proprietà *Application* permette di resituire un riferimento all'oggetto omonimo. Esempio:

```
Dim exApp as Excel.Application
```

Dichiara che la variabile oggetto *exApp* contiene un riferimento ad un oggetto *Application* di Excel. Analogamente

```
Dim woApp as Word.Application
```

Dichiara che la variabile oggetto *woApp* contiene un riferimento ad un oggetto *Application* di Word.

Oggetti specifici di Excel permettono di gestire celle, intervalli e fogli di lavoro.

- **Oggetti Workbook e Workbooks:** il primo rappresenta un file con estensione *xls* o *xla* e consente di utilizzare una sola cartella di lavoro di Excel, il secondo permette di usare tutti gli oggetti *Workbook* aperti.
- **Oggetto Worksheet:** permette di lavorare con un foglio di lavoro di Excel (che a sua volta contiene una griglia di celle).
- **Oggetto Range:** rappresenta un intervallo. Cosa sia un intervallo dipende dalle circostanze: può essere una cella o un oggetto singolo oppure un insieme di essi; può essere una riga o una colonna oppure un insieme di celle distribuite su più fogli di lavoro. Quando viene selezionata una cella (o un gruppo di esse) è possibile farvi riferimento attraverso la proprietà *Selection* (essa restituisce un oggetto di tipo *Range).*

ALCUNI ESEMPI

Vediamo ora alcuni problemi, presentando i codici sorgente dei programmi che li risolvono.

Problema 1: Prendere le righe dove esiste una cifra maggiore di una soglia prefissata e copiarle in un nuovo foglio di lavoro, chiamando tale foglio "Automatico".

Listato 2: Una possibile soluzione al problema 1.

```
1. Sub copia()
2. '
3. ' copia Macro
4. '
5. ' Scelta rapida da tastiera: CTRL+v
6. '
7.          Dim soglia As Integer
8.          Dim num As Long
9.          Dim numOrig As Long
10. Dim nuovoWs As Excel.Worksheet
11. num = 1
12. soglia = 4
13. Dim rngSelected As Range
14. Set rngSelected = Application.Selection
15. Set nuovoWs = Application.Worksheets.Add
```

Sistema

Excel e VBA
soluzioni ed esempi di codice

La tecnologia COM

COM è l'acronimo di Component Object Model. L'obiettivo di questa architettura è di creare un paradigma comune per favorire l'interazione e lo scambio di informazioni tra software differenti secondo modalità standard ed universali. Ogni applicazione è vista come l'integrazione di più componenti COM ad ognuno dei quali competono delle funzionalità.

Sistema

Excel e VBA
soluzioni ed esempi di codice

```
16. nuovoWs.Name = "Automatico"
17. For Each rngTmp In rngSelected
18. If (rngTmp.Value = "") Then
19.         Exit Sub
20. ElseIf (rngTmp.Value > soglia) Then
21.         nuovoWs.Range("A" & num).Value =
                                          rngTmp.Value
22.         num = num + 1
23. End If
24. Next rngTmp
25. End Sub
```

Microsoft Excel - vba.xls

	A	B	C	D
1	Questa è la prima riga [re34543]	10/10/2001		2
2	Seconda riga d'esempio [656]	12/10/1999		3
3	Terza riga	1/2/2002		8
4	Riga numero quattro [45654]	1/7/2001		5
5	Quinta riga? [Si] [6564]	12/12/2001		6
6	Sesta riga [64545]	30/1/2002		8
7				

Fig. 3: **Il foglio dilavoro di partenza, su cui eseguiremo le varie macro presentate.**

Analizziamo le istruzioni "importanti": *(15)* si memorizza un riferimento alla selezione attuale; *(16)* viene creato un nuovo foglio di lavoro a cui viene dato il nome di *"Automatico"*; *(17)* si esaminano tutte le celle selezionate; *(18-19)* se si trova una cella vuota, si esce; *(20-24)* altrimenti, se la cella contiene un valore maggiore della soglia prefissata, si scrive tale valore in una nuova cella del foglio di lavoro *"Automatico"*.

Problema 2: Sulla colonna selezionata, mettere in evidenza (per esempio colorando lo sfondo) le date maggiori di un mese, 6 mesi, 1 anno della data attuale (con colori diversi, rispettivamente verde, azzurro, rosso).

Listato 3: Ecco una possibile seoluzione al problema 2.

```
1. Sub colora()
2. '
3. ' colora Macro
4. '
5. ' Scelta rapida da tastiera: CTRL+c
6. '
7.      Dim today As Date
8.      Dim colora As Boolean
9.      Dim colore As Integer
10. today = Now
11. For Each rngTmp In Application.Selection
12.     If (rngTmp.Value = "") Then
13.         Exit Sub
14.     End If
15.     colora = False
16.     colore = 0
17.     If (Abs(DateDiff("yyyy", rngTmp.Value,
                                          today)) > 1) Then
18.         colore = 3
19.         colora = True
20.     ElseIf (Abs(DateDiff("m", rngTmp.Value,
                                          today)) > 6) Then
21.         colore = 8
22.         colora = True
23.     ElseIf (Abs(DateDiff("m", rngTmp.Value,
                                          today)) > 1) Then
24.         colore = 4
25.         colora = True
26.     End If
27.     If (colora) Then
28.         rngTmp.Select
29.         With Selection.Interior
30.                 .ColorIndex = colore
31.                 .Pattern = xlSolid
32.                 .PatternColorIndex = xlAutomatic
33.         End With
34.     End If
35. Next rngTmp
36. End Sub
```

Microsoft Excel - vba.xls

	A	B	C	D
1	Questa è la prima riga [re34543]	10/10/2001		2
2	Seconda riga d'esempio [656]	12/10/1999		3
3	Terza riga	1/2/2002		8
4	Riga numero quattro [45654]	1/7/2001		5
5	Quinta riga? [Si] [6564]	12/12/2001		6
6	Sesta riga [64545]	30/1/2002		8
7				

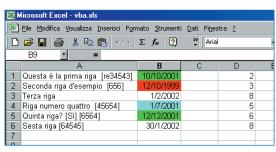



Fig. 4: **La macro "colora" eseguita in data 16 febbraio 2002, dopo aver selezionato la colonna "B".**

Osserviamo alcune linee del listato: *(10)* la funzione *Now* restituisce la data attuale (nel momento in cui viene eseguito il programma; *(11)* il ciclo avviene per ogni cella della selezione; *(12-14)* se ha raggiunto una cella vuota esce; *(17, 20, 23)* viene usata la funzione *DateDiff* per eseguire la differenza tra due date.

Problema 3: Supponiamo di aver memorizzato dei codici alla fine di una etichetta. Tali codici sono racchiusi tra parentesi quadre: *"nome etichetta [codice]"*. Vogliamo leggere i codici, eliminarli dalla colonna, lasciandovi solo l'etichetta, e riscriverli in una colonna a sé stante.

Listato 4: Soluzione al problema 3.

```
1. Option Explicit
2. Sub dividiCodice()
3. '
4. ' dividiCodice Macro
5. '
6. ' Scelta rapida da tastiera: CTRL+x
7. '
```

Come proteggere una Macro

Dopo aver creato una macro, si può evitare che altri utenti possano accedere al suo contenuto semplicemente inserendo una password di apertura. Per far questo, dal menu Strumenti, si sceglie la voce Macro/ Visual Basic Editor :verrà visualizzato l'editor VBA. Dal menu Strumenti, si seleziona la voce Strumenti/Proprietà di VBA Project. Nella scheda Protezione è possibile inserire una password e proteggere l'accesso alla macro.

Sistema

Excel e VBA
soluzioni ed esempi di codice

```
8.     Dim rng As Excel.Range
9.     Dim indice As Long
10. Dim rngTmp As Range
11. Dim divChar As String
12. divChar = "["
13. For Each rngTmp In Application.Range("A:A")
14.     If (rngTmp.Value = "") Then
15.                 Exit Sub
16.     End If
17.     Dim matrice() As String
18.     Dim st As String
19.     st = rngTmp.Value
20.     matrice = Split(st, divChar)
21.     indice = UBound(matrice)
22.     If (indice > 0) Then
23.         st = divChar & matrice(indice)
24.         rngTmp.Offset(0, 2).Value = st
25.         Dim i As Long
26.         For i = LBound(matrice) To indice - 2
27.                 matrice(i) = matrice(i) & divChar
28.         Next
29.         matrice(indice) = ""
30.         rngTmp.Value = Join(matrice)
31.     End If
32. Next rngTmp
33. End Sub
```

Analizziamo la soluzione proposta: *(13)* il ciclo avviene prendendo in considerazione la prima colonna; *(20)* viene usata la funzione *Split*: essa prende due argomenti (due stringhe): considera la prima stringa e la divide usando come elemento separatore la seconda stringa. Ritorna una matrice di stringhe (Esempio *Split("a#b#c","#") ‡ ["a", "b", "c"]); (22)* se non esiste almeno una occorrenza del carattere separatore, allora *(32)* va alla prossima istruzione del ciclo, altrimenti *(23)* ricompone l'ultima parte della stringa con il carattare separatore e *(24)* lo assegna alla cella che si trova due colonne dopo la cella attuale *(Offsset(riga, colonna)* rispetto a quella attuale); *(26-28)* ricompone le stringhe precedenti (nel caso esistessero altre parti della stringa composte dal carattere separatore preso in esame); (30) esegue l'operazione di Join tra stringhe: essa è l'opposto dell'operazione di *Split* (Esempio *Join(["a", "b", "c"]) ‡ "abc")*. Anziché copiare i valori su una colonna esistente, sarebbe possibile inserire

Microsoft Excel - vba.xls

File Modifica Visualizza Inserisci Formato Strumenti Dati Finestra ?

B19

	A	B	C	D
1	Questa è la prima riga	10/10/2001	[re34543]	2
2	Seconda riga d'esempio	12/10/1999	[656]	3
3	Terza riga	1/2/2002		8
4	Riga numero quattro	1/7/2001	[45654]	5
5	Quinta riga? [Sì]	12/12/2001	[6564]	6
6	Sesta riga	30/1/2002	[64545]	8
7				

Fig. 5: **Il risultato della macro dividiCodice() (Listato 4).**

una nuova colonna. Come? Per esempio in questo modo:

```
Columns("B:B").Select
Selection.Insert Shift:=xlToRight
```

La prima istruzione seleziona la seconda colonna, la seconda istruzione inserisce una nuova colonna spostando (*shift*) a destra il resto del foglio di lavoro.

INFORMAZIONI IN RETE

Spesso è utilissimo avere a portata di mano della documentazione per risolvere i problemi più comuni. Per questo, diventa di fondamentale importanza trovare buoni siti Web, contenenti articoli , tutorial o semplici tips. Tra essi segnalo il sito della Microsoft per i prodotti Office (*http://office.microsoft.com/*), e il sito ITPortal, di Edizioni Master, nella sezione dedicata a Visual Basic (*http://www.itportal.it/developer/vb/*). Per esempi e brevi macro potete andare alle pagine: *"Chris Rae's VBA pages"* (*http://chrisrae.com/vba/routines.html*), "Microsoft Excel VBA Examples" (*http://www.mindspring.com /~tflynn/excelvba.html*), *"Word-VBA Code Samples"* (*http://www.jojo-zawawi.com/code-samples-pages/code-samples.htm*). Oltre ai siti segnalati potete usare il vostro motore di ricerca preferito e ricercare i documenti che possono risolvere i problemi che via via vi si possono presentare. Uno vantaggi di Visual Basic *for Applications* è che esso è diventato lo standard de facto nella personalizzazione di applicazioni Windows: non solo i prodotti Microsoft, ma molti altri prodotti offrono la possibilità di aggiungere funzionalità in tale linguaggio. Pertanto lo sforzo iniziale per impararlo sarà ripagato nel lungo termine, potendo sfruttare le conoscenze acquisite in molte circostanze.
L'ultimo consiglio è quello di imparare ad usare molto bene le applicazioni prima di accingervi a scrivere delle macro: se non conoscete le funzionalità "operative" dello strumento è molto difficile che riusciate a farne un uso efficace nei vostri programmi.

CONCLUSIONI

Con questo articolo si è voluto dare un "assaggio" delle caratteristiche del VBA applicate ad Excel. L'invito è quello di partire dagli esempi proposti per creare nuove funzionalità, approfondendo la propria conoscenza sia di VBA che di Excel. Nei prossimi mesi avremo modo di trattare anche altri argomenti legati alla programmazione VBA nei pacchetti Office: in modo particolare vedremo com'è possibile usare singole macro che facciano uso di funzionalità provenienti da pacchetti diversi (Outlook, Word, Excel, Power Point e così via).

Ivan Venuti

Bibliografia

- **MANUALE DELLLO SVILUPPATORE DI MICROSOFT OFFICE XP DEVELOPER** *Microsoft Press* (Mondadori Informatica) 2001

Sistema

C++ e Delphi

COME UTILIZZARE I COMPONENTI DELPHI IN C++

Accade spesso che si abbia a disposizione un componente, o una parte di codice che vorremmo riutilizzare in una applicazione scritta in un linguaggio diverso. È possibile, evitando la traduzione da un linguaggio all'altro, riutilizzare il codice già pronto e collaudato da Delphi a C++ e viceversa: vediamo come questo sia possibile.

File sul CD
\soft\codice\component.zip

Programmazione mista Delphi e C++

La tecnica descritta in questo articolo permette di utilizzare un componente Delphi all'interno di una applicazione C++, con indubbi vantaggi dal punto di vista di ottimizzazione delle risorse e del codice disponibile da parte del programmatore.

Quante volte abbiamo trovato un componente che ci sarebbe risultato estremamente utile, ma che sfortunatamente era scritto in un linguaggio diverso da quello che utilizziamo comunemente: al sottoscritto questa eventualità è capitata numerose volte.
Fortunatamente, nella mia 'cassetta degli attrezzi' informatica ho due linguaggi di programmazione potenti e versatili, pur nelle loro diversità e peculiarità, che mi consentono di riutilizzare il codice scritto in un linguaggio, con poche e semplici operazioni nell'altro.
In queste pagine vedremo come sfruttare un componente Delphi, inserendolo in una applicazione C++, completa e funzionante: ovviamente codice e componenti sono reperibili facilmente nel CD allegato alla rivista.

Fig. 1: **Dopo l'installazione il componente Delphi viene reso disponibile sulla barra dei componenti, come un qualunque altro modulo C++.**

PERCHÉ UTILIZZARE COMPONENTI DELPHI IN C++

La prima domanda che ci si pone è perché si debba utilizzare una parte di codice scritta in un altro linguaggio, invece di produrre un programma scritto interamente nel nostro linguaggio preferito. Le motivazioni possono essere innumerevoli: innanzi tutto considerazioni di tempo e costi di produzione, conviene sicuramente inserire un componente già pronto piuttosto che scriverne uno dall'inizio. Pensiamo soltanto ai componenti di gestione delle linee di comunicazione, per i quali sono necessari mesi di sviluppo ed una mole notevole di risorse per verifica e debug. In aggiunta a ciò possiamo dire che è senz'altro più sicuro l'utilizzo di un componente già collaudato, piuttosto che sobbarcarsi l'impresa di scrittura e debug di un nuovo codice. Potremmo dire che oltre alle considerazioni fatte finora, si ha la possibilità di attingere a più fonti di librerie, che possono facilitarci la ricerca del materiale che ci può servire. Un aspetto importante inoltre riguarda gli studenti: spesso viene utilizzato il Pascal, nella maggior parte dei corsi universitari, come linguaggio di programmazione iniziale, per le sue doti didattiche indiscusse e quindi spesso il Delphi come naturale evoluzione. La migrazione dal Pascal al C ed al C++, può risultare per certi versi frustrante, ma necessaria per motivi di maggiore diffusione di questo ultimo linguaggio: al di là delle differenze dei vari ambienti, che esulano dallo scopo di queste pagine, resta il disappunto di non potere riutilizzare il codice scritto tanto faticosamente in Pascal. Personalmente utilizzo molto Delphi, ma anche C++, a seconda del tipo di applicazione che devo produrre, al primo attribuisco una eccezionale velocità produttiva e resta comunque la mia prima passione; al secondo devo riconoscere una maggiore diffusione ed una elevata quantità di librerie disponibili: in definitiva posso asserire che il loro utilizzo congiunto possa essere una carta vincente in moltissime occasioni.

IL COMPONENTE DELPHI

A titolo di esempio proponiamo un semplice componente Delphi, che ha lo scopo di simulare, sotto forma grafica, un diodo LED, che abbia la caratteristica di potere essere ridimensionato e ridefinito nelle proprie caratteristiche fondamentali, ed in particolare nella pro-

Sistema

C++ e Delphi

Come utilizzare i Componenti Delphi in C++

pria forma e colore. Vogliamo che il LED possa essere acceso per mezzo di una procedura, chiamata *LedON* e spento analogamente attraverso *LedOFF*, nell'ottica di una buona programmazione ad oggetti, evitando di rendere disponibili all'utente direttamente le variabili di controllo private. Si può verificare se il LED è acceso o meno attraverso la proprietà *SpuntoLedisON*, mentre i colori da attribuire al LED durante lo stato di 'acceso' e 'spento', vengono impostati rispettivamente attraverso le proprietà *SpuntoLedOnColor* e *SpuntoLedOffColor*. Come si può notare dall'immagine che segue, le proprietà del componente possono essere modificate attraverso l'Object Inspector, direttamente in fase di progettazione della form, come è già stato detto, per l'accensione e lo spegnimento occorre utilizzare le apposite procedure. Per creare un gestore degli eventi, basta selezionare la tag *Events* dell'Object Inspector ed una volta individuato l'evento in questione è necessario fare doppio click sulla linea relativa per forzare la creazione dello scheletro della procedura che funge da 'Event Handler'. Nel nostro esempio vogliamo che quando l'utente posiziona il cursore sull'oggetto Led, venga forzata l'esecuzione di una procedura che accenda il Led stesso, come vedremo velocemente più avanti. Veniamo ora all'analisi del componente Delphi: per motivi didattici l'applicazione ed il componente sono molto semplici, ma possiedono tutte le funzionalità per descrivere le tecniche di utilizzo dei due linguaggi che ci siamo prefissati. Innanzi tutto l'intestazione comprende una breve descrizione del componente, di seguito figurano il nome della unit che lo contiene ed a sua volta, nella clausola *uses* troviamo le units che il componente utilizza.

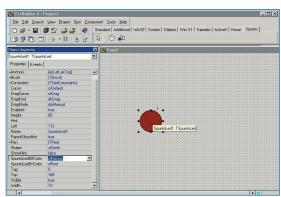
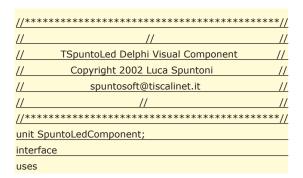



Fig. 2: Le proprietà del componente possono venire modificate utilizzando l'***Object Inspector***.

```
//*********************************************//
//                    //                      //
//      TSpuntoLed Delphi Visual Component    //
//        Copyright 2002 Luca Spuntoni        //
//           spuntosoft@tiscalinet.it         //
//                  //                        //
//*********************************************//
unit SpuntoLedComponent;
interface
uses
```

```
  Windows, Messages, SysUtils, Classes, Controls,
                               ExtCtrls, Graphics;
```

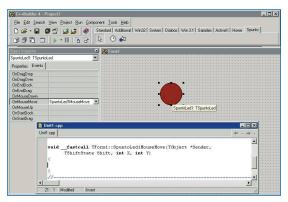
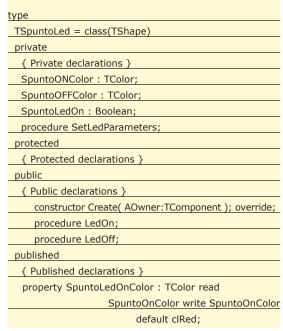



Fig. 3: Per generare lo scheletro del codice di un gestore di eventi, occorre selezionare la parte 'events' dell'***Object Inspector***, per poi fare doppio clic sull'evento desiderato.

Nella intestazione dei tipi troviamo la descrizione dell'unica classe che si trova all'interno della unit: come si può notare fin da una prima analisi, il nostro LED discende dalla classe *TShape*, che ingloba già molte delle funzionalità che possono servirci; osserviamo poi nella parte *Private* le variabili che contengono i colori da assegnare al LED al momento della sua accensione e del suo spegnimento, nonché il suo stato logico, interpretabile rispettivamente come *True=Acceso* e *False=Spento*. L'unica procedura che si trova nella parte *Private* è *SetLedParameters*, che provvede ad impostare i parametri del LED, come vedremo nel dettaglio più avanti. Nella sezione *Public* possiamo notare le due procedure che provvedono all'accensione ed allo spegnimento del LED, dal significato abbastanza ovvio, mentre nella parte *Published*, oltre al costruttore apprezziamo finalmente le proprietà relative ai colori del LED durante lo stato di acceso e di spento, nonché *SpuntoLedIsOn*, che restituisce lo stato logico del componente all'utente.

```
type
  TSpuntoLed = class(TShape)
  private
    { Private declarations }
    SpuntoONColor : TColor;
    SpuntoOFFColor : TColor;
    SpuntoLedOn : Boolean;
    procedure SetLedParameters;
  protected
    { Protected declarations }
  public
    { Public declarations }
      constructor Create( AOwner:TComponent ); override;
      procedure LedOn;
      procedure LedOff;
  published
    { Published declarations }
    property SpuntoLedOnColor : TColor read
                         SpuntoOnColor write SpuntoOnColor
                                 default clRed;
```

Il Componente Delphi

Il componente proposto in queste pagine consente di simulare il comportamento di un LED, comunemente utilizzato in elettronica.

Sistema

C++ e Delphi
Come utilizzare i Componenti Delphi in C++

```
  property SpuntoLedOffColor : TColor read SpuntoOffColor
                          write SpuntoOffColordefault clMaroon;
  property SpuntoLedIsON: Boolean Read SpuntoLedOn;
 end;
```

La procedura *Registrer*, dichiarata prima al termine della sezione *Interface* e poi definita nella parte *Implementation* si occupa di indicare il nome del componente da registrare sulla barra dei componenti e stabilisce in quale segnaposto debba essere inserito.

```
procedure Register;
implementation
procedure Register;
begin
  RegisterComponents('Spunto', [TSpuntoLed]);
end;
```

Il costruttore del componente provvede innanzi tutto ad eseguire tutte lo operazioni contenute nel costruttore della classe progenitrice, attraverso l'istruzione *Inherited*, successivamente imposta i parametri peculiari del nostro LED, in termini di colore, stato di accensione, dimensioni e forma del componente stesso: in aggiunta a ciò provvede, attraverso la procedura *SetLedParameters* a colorare il LED con il colore stabilito a seconda che il componente sia acceso oppure spento.

```
constructor TSpuntoLed.Create( AOwner : TComponent );
begin
  inherited Create( AOwner );
  SpuntoONColor :=clRed;
  SpuntoOFFColor:=clMaroon;
  SpuntoLedOn   :=False;
  Shape         :=stcircle;
  Width         := 15;
  Height        := 15;
  SetLedParameters;
end;
```

Nel dettaglio la procedura *SetLedParameters*, provvede ad associare il colore del componente a *SpuntoOnColor*, qualora sia acceso ed a *SpuntoOffColor*, nell'eventualità che sia spento.

```
procedure TSpuntoLed.SetLedParameters;
Begin
 If SpuntoLedOn then Brush.Color:=SpuntoOnColor else
                            Brush.Color:=SpuntoOffColor;
End;
```

L'analisi di questo semplicissimo componente giunge al termine esaminando le due procedure atte all'accensione ed allo spegnimento del Led. Si è preferito utilizzare questo metodo, anziché permettere all'utente di accedere o modificare il valore logico di una variabile interna per motivi inerenti alla buona programmazione ad oggetti, inoltre un componente di questo genere è più adatto all'impiego in programmi destinati alla gestione di apparecchiature elettroniche.

```
procedure TSpuntoLed.LedOn;
Begin
 SpuntoLedOn:=True;
 SetLedParameters;
End;
procedure TSpuntoLed.LedOff;
Begin
 SpuntoLedOn:=False;
 SetLedParameters;
End;
end.
```

Installazione del Componente
L'installazione del componente può essere effettuata facilmente selezionando *Component/Install Component*, utilizzando C++ Builder.

INSTALLAZIONE DEL COMPONENTE DELPHI IN C++

Una volta terminata l'analisi del componente, vediamo come sia possibile installarlo nell'ambiente integrato di C++ Builder. L'installazione è molto semplice: dopo avere selezionato *Component/Install Component*, occorre selezionare un package nel quale si voglia installare il componente in questione, a questo punto richiamo la unit appropriata, nel nostro caso *SpuntoLedComponent.Pas* e lanciamo l'installazione. Sarà possibile a questo punto, aprendo il package analizzarne la struttura ed eventualmente accedere con un doppio click alle caratteristiche dei componenti in esso installati.

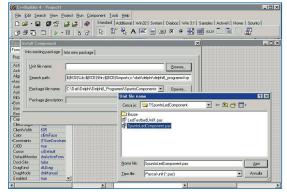



Fig. 4: Per installare un componente Delphi in C++, utilizzando C++ Builder è sufficiente selezionare *Component/Install Component*, scegliere il Package appropriato ed infine selezionare il nome della unit Delphi desiderata.

L'APPLICAZIONE C++ DOTATA DI COMPONENTE DELPHI

Dopo avere creato il componente ed averlo installato, analizziamo l'utilizzazione del codice Delphi, contenuto nella classe *TSpuntoLed* da parte di un programma C++. L'applicazione al momento dell'esecuzione viene mostrata nella figura che segue: nella parte centrale della finestra appare il nostro grosso LED, istanza del componente Delphi che abbiamo creato ed installato in C++ Builder; appaiono inoltre altri due piccoli LED sul

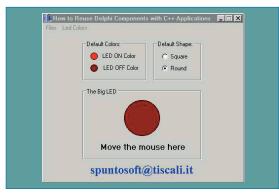



Fig. 5: **Lanciando l'applicazione creata fino a questo punto, si ottiene la schermata di figura.**

lato sinistro, che hanno lo scopo di mostrare il colore del componente quando si trova nello stato 'acceso' e 'spento'. Gli oggetti rimanenti che si trovano sulla form sono componenti nativi C++: ci troviamo quindi di fronte ad una forma interessante di applicazione 'ibrida', mista cioè C++ e Delphi. Analizzando il codice dell'applicazione, possiamo notare dopo l'intestazione di rito, le direttive di inclusione dei file header del preprocessore: in modo particolare facciamo caso alla direttiva *#include "SpuntoCppLedUnit1.h"*, che permette l'inserimento del codice relativo al componente che abbiamo appena analizzato ed installato. Riferendoci sempre al componente Delphi, notiamo la direttiva *#pragma link "SpuntoLedComponent"*, la quale indica al linker di inserire il componente in questione nel file eseguibile. Si possono notare inoltre le altre direttive che vengono generate automaticamente dall'IDE, compresa la definizione della form principale.

```
//******************************************//
//            //              //            //
// How to Reuse Delphi Components with C++
                               Applications //
//      Copyright 2002 Luca Spuntoni        //
//         spuntosoft@tiscali.it            //
//            //            //              //
//******************************************//
//-----------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "SpuntoCppLedUnit1.h"
//-----------------------------------------------------------
#pragma package(smart_init)
#pragma link "SpuntoLedComponent"
#pragma resource "*.dfm"
TSpuntoLedDelphitoCPPApplication
*SpuntoLedDelphitoCPPApplication;
//-----------------------------------------------------------
__fastcall TSpuntoLedDelphitoCPPApplication::
        TSpuntoLedDelphitoCPPApplication(TComponent*
                                Owner): TForm(Owner)
{}
```

Nella form principale è stato inserito un menu principale, dotato di un item deputato alla chiusura della finestra: la funzione *Exit1Click*, si preoccupa di distruggere l'istanza del programma, come viene riportato di seguito e come è facilmente comprensibile.

```
//-----------------------------------------------------------
void __fastcall TSpuntoLedDelphitoCPPApplication::
                              Exit1Click(TObject *Sender)
{Close();        //Exits the Application }
```

Abbiamo detto che vogliamo che sia possibile cambiare i colori del LED, sia nello stato di 'acceso' che di 'spento': le procedure che seguono si occupano proprio di questo assegnando al led grande, individuato da *SpuntoBigLed* il colore scelto dall'utente attraverso il dialogo di scelta dei colori *ColorDialog1*. Oltre a questo i due LED più piccoli *SpuntoLed1* e *SpuntoLed2*, che individuano i LED di esempio rispettivamente per lo stato di 'acceso' e di 'spento' vengono posti nello stato logico corrispondente, una volta che gli è stato assegnato il colore appropriato.

```
//-----------------------------------------------------------
void __fastcall TSpuntoLedDelphitoCPPApplication::
                         ONColor1Click(TObject *Sender)
{ if (ColorDialog1->Execute())     // LED ON COLOR
 SpuntoLed1->SpuntoLedOnColor = ColorDialog1->Color;
 SpuntoBigLed->SpuntoLedOnColor = ColorDialog1->Color;
 SpuntoLed1->LedOn(); }
//-----------------------------------------------------------
void __fastcall TSpuntoLedDelphitoCPPApplication::
                        OFFColor1Click(TObject *Sender)
{ if (ColorDialog1->Execute())       // LED OFF COLOR
 SpuntoLed2->SpuntoLedOffColor = ColorDialog1->Color;
 SpuntoBigLed->SpuntoLedOffColor = ColorDialog1->Color;
 SpuntoLed2->LedOff(); }
```

Alla creazione della finestra viene eseguito il codice che segue, semplicemente impostando i LED di esempio allo stato di 'acceso e 'spento', attraverso le procedure *LedOn* e *LedOff*, oltre a definire il LED grande di forma rettangolare con gli spigoli arrotondati (*stRoundRect*).

```
//-----------------------------------------------------------
void __fastcall TSpuntoLedDelphitoCPPApplication::
                          FormCreate(TObject *Sender)
{ SpuntoLed1->LedOn();             // LED SETUP
  SpuntoLed2->LedOff();          // AT FORM CREATION
  SpuntoBigLed->Shape= stRoundRect;}
```

Quando l'utente muove il cursore del mouse sopra al LED grande, viene generato un evento *MouseMove* da parte dell'istanza *SpuntoBigLed* e quindi viene eseguita la procedura che segue. In effetti viene sfruttata la procedura soltanto come gestore dell'evento determinato dal posizionamento del mouse sull'oggetto: non sono utilizzati gli altri parametri quali ad esempio la posizione del mouse nelle coordinate *X,Y* che potrebbe

Sistema

C++ e Delphi
Come utilizzare i Componenti Delphi in C++

Applicazione ibrida
L'applicazione può definirsi 'ibrida', in quanto ingloba all'interno di una applicazione C++, una serie di istanze di un componente nativo Delphi.

Sistema

C++ e Delphi
Come utilizzare i Componenti Delphi in C++

aprire il campo ad espansioni molto interessanti che vengono lasciate al lettore. In definitiva quando l'utente posiziona il cursore sul LED, quest'ultimo viene acceso.

```
//------------------------------------------------------------
void __fastcall TSpuntoLedDelphitoCPPApplication::
        SpuntoBigLedMouseMove(TObject *Sender,
                     TShiftState Shift, int X, int Y)
{ SpuntoBigLed->LedOn();  // BIG LED ON WHEN
                                   MOUSE MOVE }
```

Per rendere possibile lo spegnimento del LED sfruttiamo il gestore dell'evento *OnFormMouseMove* che si verifica quando il cursore viene spostato nell'area della form non compresa dalla superficie del LED, in questo caso quest'ultimo viene spento attraverso l'istruzione *SpuntoBigLed->LedOff();*.

```
//------------------------------------------------------------
void __fastcall TSpuntoLedDelphitoCPPApplication::
     FormMouseMove(TObject *Sender, TShiftState Shift,
     int X, int Y)
{ SpuntoBigLed->LedOff();    // BIG LED OFF WHEN
                            OUTSIDE } // LED AREA
//------------------------------------------------------------
```

Potenza e flessibilità

In definitiva l'utilizzo congiunto delle potenzialità di Delphi e di C++ conferisce potenza e flessibilità alle applicazioni, potendo sfruttare i punti di forza dei due ambienti di programmazione.

La gestione della forma del LED avviene attraverso le procedure riportate appresso, mediante le quali si modifica la proprietà *shape* del LED, utilizzando le costanti standard *stRoundRect* e *stCircle*, corrispondenti rispettivamente ad un rettangolo con bordi arrotondati ed ad un cerchio: il tutto avviene attraverso la selezione di uno dei due radiobutton corrispondenti alla forma desiderata dall'utente.

```
void __fastcall TSpuntoLedDelphitoCPPApplication::
          RadioButtonSquareClick(TObject *Sender)
{ SpuntoBigLed->Shape= stRoundRect;
                         // LED SHAPE ROUND RECT}
//------------------------------------------------------------
void __fastcall TSpuntoLedDelphitoCPPApplication::
          RadioButtonRoundClick(TObject *Sender)
{ SpuntoBigLed->Shape= stCircle; // LED SHAPE CIRCLE}
//------------------------------------------------------------
```

Fig. 6: **Posizionando il cursore del mouse sul LED, viene forzata l'esecuzione della parte di codice corrispondente al gestore dell'evento 'OnMouseMove'.**

L'analisi del codice C++ è terminata, si è potuto notare come sia possibile accedere a tutte le funzionalità del nostro componente Delphi anche attraverso l'utilizzo di C++ e come sia stato possibile riutilizzare il codice che è in nostro possesso, evitando la traduzione e la conseguente 'ri-scrittura' del componente.

UTILIZZO DELL'APPLICAZIONE

Lanciata l'applicazione inclusa nel CD allegata alla rivista, oppure 'ricompilandola' attraverso C++ Builder si ottiene la schermata della figura che segue: posizionando il cursore del mouse sul LED grande, si ottiene l'accensione del componente stesso, per mezzo dei meccanismi che sono stati descritti in precedenza.
Nel caso si voglia modificare il colore del LED sia durante il suo stato di 'acceso', che di 'spento', è sufficiente selezionare l'opzione corrispondente del menu *Led Colors*, come si può notare dall'immagine nella figura seguente per modificare i colori del componente.
L'utilizzo dell'applicazione nelle sue funzioni di base è molto semplice e non necessita di ulteriori commenti.
Tutto il codice presentato in queste pagine, il componente Delphi e l'applicazione C++ completa e funzionante è reperibile sul CD nel file : *SpuntoCPP_Delphi_Component.zip*.

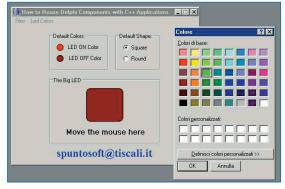



Fig. 7: **L'applicazione permette di modificare il colore del LED, sia quando questo è acceso, che quando è spento, oltre a renderne possibile la visualizzazione in forma sia circolare che rettangolare.**

CONCLUSIONI

In queste pagine abbiamo visto come inserire ed utilizzare un componente Delphi creato precedentemente, oppure reso disponibile attraverso l'acquisizione attraverso terze parti, in una applicazione C++. Abbiamo parlato dei vantaggi di questa tecnica, in particolare per coloro che utilizzano indifferentemente questi due potenti linguaggi di programmazione, oppure per quella grossa parte di studenti universitari che hanno in questo modo l'opportunità di riutilizzare il proprio codice Object Pascal. L'autore è felice di rispondere alle domande ed accettare i suggerimenti dei lettori all'indirizzo di posta elettronica: *spuntosoft@tiscali.it*.

Luca Spuntoni

WinHelp

INTEGRARLO IN UN PROGETTO

In un'applicazione di buon livello non può mancare una sezione dedicata alla spiegazione dell'applicazione stessa, proprio in quest'ottica, descriveremo come implementare un Help e come agganciarlo ad un progetto Visual Basic.

Sicuramente il modo migliore per documentare un'applicazione è quello di corredarla di una guida in linea multimediale, in altre parole di una guida simile a quella dei sistemi operativi Windows che come è noto devono il loro successo, anche, all'interfaccia amichevole e ben documentata.
La guida in linea per un'applicazione Visual Basic può essere di due tipi:

1. **WinHelp**, costruita interamente con tecnologie Windows;

2. **HTML Help**, costruita utilizzando la tecnologia HTML.

Le guide HTML sono più moderne ed efficienti ed ormai hanno soppiantato le WinHelp.
Sul mercato attualmente sono disponibili vari pacchetti di terze parti per l'implementazione e la distribuzione di guide in linea, tra tutti citiamo RoboHelp e l'onnipresente Flash MX.
Naturalmente in questo articolo non faremo degli esempi basati su questi prodotti ma utilizzeremo degli strumenti già in vostro possesso o reperibili, facilmente, su Internet. Gli strumenti in questione sono: Winhelp (in realtà dovremmo parlare di *HCP.exe* come sarà chiaro tra poco) e *HTML Help Workshop 4.74*. Essi sostanzialmente sono dei compilatori che ricevono in input dei file opportunamente strutturati e producono in output i file della guida, da agganciare ad un progetto.
In particolare il Winhelp è un compilatore che lavora soltanto in MsDos mentre il Workshop è un vero ambiente di sviluppo e lavora anche in Windows XP. Nel corso dell'articolo descriveremo:

1. i compilatori WinHelp e HTMLHelp ed i loro file di input;

2. come agganciare un file di Help ad un'applicazione Visual Basic;

3. il Wizard del Workshop HTML, che da un progetto WinHelp permette di ottenere un progetto HTML Help.

Come esempio presenteremo un'applicazione che permette di gestire i dati dei dipendenti di un'azienda.

LA GUIDA IN LINEA

In generale un progetto di una guida in linea è composto: da un insieme di pagine (denominate Topics-argomenti); da un file di contesto (context file) che rappresenta gli argomenti principali in una rappresentazione ad albero; da un file d'indice (Index file) che contiene l'elenco delle chiavi di ricerca; da un insieme di file o comandi (dipende dal tipo di guida) che contengono informazioni sui Link tra i vari Topics e con i POPUP (i Popup sono dei messaggi corti da agganciare ai singoli oggetti dell'interfaccia, per capirci sono simili ai ToolTipText) ed un insieme di file di supporto (immagini, video e sonori).
I Topics nel caso di WinHelp sono definiti attraverso un file RTF mentre nel caso HTML possono essere definiti come semplici file HTML.
Le informazioni sui vari file dell'Help, sono organizzate in un file di progetto che nel caso di WinHelp è un file in formato ASCII con estensione HPJ, mentre nel caso dell'HMTL Help è un file del Workshop con estensione HHP.

MICROSOFT WINHELP

Una guida sviluppata in formato Help di Windows è un file compilato con estensione *HLP.* Il compilatore può essere uno dei seguenti: HC.EXE, HC31.EXE, HCP.EXE e HCRTF.EXE. In particolare HCRTF.EXE produce un Help file a 32 bit (per i sistemi operativi Windows 95 o superiori).
Per i nostri esempi abbiamo utilizzato il compilatore HCP.EXE che funziona solo in modalità MSDOS per questo la compilazione può essere avviata solo attra-

Visual Basic

Formati

Le informazioni sui vari file dell'Help, sono organizzate in un file di progetto che nel caso di WinHelp è un file in formato ASCII con estensione HPJ, mentre nel caso dell'HMTL Help è un file del Workshop con estensione HHP.

Visual Basic

verso un comando MSDOS. Ricordiamo che un comando MSDOS può essere inviato: attraverso il Prompt MSDOS, attraverso un file BAT oppure attraverso il programma "Esegui" di Windows. Per questi tipi di compilatori, i Topics sono organizzati in un file RTF, il file di progetto è un HPJ, mentre il contesto è organizzato in un file con estensione CNT, come illustreremo tra poco.

MICROSOFT HTML HELP

Una guida sviluppata in formato HTML Help è un file compilato con estensione CHM.
Microsoft HTML Help è un ambiente che permette di implementare guide in linea utilizzando le caratteristiche dell'HTML, quindi delle guide che possono essere agganciate ad un progetto, sviluppato con Visual Studio, oppure pubblicate su Internet. L'ambiente è composto da un Help Viewer, dagli Help components e da alcuni tools (come Help Image Editor). L'Help Viewer basa il suo funzionamento sui componenti di Internet Explorer (in particolare su *Shdocvw.dll*) per questo è compatibile con la maggior parte dei sistemi operativi della Microsoft. Esso supporta: HTML, ActiveX, Java, JScript, VbScript ed i formati immagini: jpeg, gif, png.
Un progetto HTML Help per molti aspetti è organizzato come un progetto WinHelp, per questo il Workshop fornisce un Wizard che consente di convertire i progetti WinHelp in progetti HTML Help, anche se bisogna preparare qualche accorgimento soprattutto per la conversione degli elementi contestuali: Popup e albero di contesto. Nel nostro esempio, considereremo prima un progetto WinHelp e poi con il Wizard lo convertiremo in un progetto HTML Help che, infine, dopo la compilazione aggancerемо ad un progetto Visual Basic. Allora prima introduciamo il progetto Visual Basic e poi descriviamo come sviluppare la sua guida in linea.

Help image

Microsoft HTML Help Image Editor 4.0 è un tool per gestire le immagini. Esso può essere attivato direttamente dall'IDE del Workshop Help e permette di fare varie operazioni sulle immagini, per esempio: conversione di tipi, modifica, organizzazione in album, capture. Naturalmente le immagini gestite con Help Image, con semplici passaggi, possono essere inserite nelle pagine (Topics) dell'Help.

IL PROGETTO VISUAL BASIC

L'applicazione di esempio permette di gestire i dati anagrafici dei dipendenti di un'azienda. Essa dovrebbe avere un certo numero di form e un database di supporto. Noi, invece, svilupperemo l'essenziale cioè solo la maschera (form) che permette d'inserire i dati anagrafici dei dipendenti. La Form in questione contiene una *SSTab* con tre schede che per semplicità nominiamo: *Prima*, *Seconda* e *Terza Pagina*. Su ogni scheda inseriremo un certo numero di textbox ed etichette.
In particolare nella form inseriremo solo il codice che permette di gestire, attraverso alcune procedure, il dialogo con la guida in linea, mentre in un modulo di supporto, oltre alle citate procedure, inseriremo la definizione delle funzioni dell'API (cioè dell'HTML Help API) che consentono, ad un progetto Visual Basic, di interagire con gli elementi della guida.

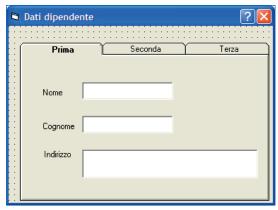



Fig. 1: La form in fase di progettazione

Per quanto riguarda la guida, in essa definiremo un Topic per ogni scheda della *SStab* e un messaggio POPUP per ogni oggetto presente sulle schede (quest'ultima parte la descriveremo nel successivo articolo).
Prima di descrivere come implementare la guida, vediamo che cosa bisogna predisporre in un progetto Visual Basic per poterlo agganciare ad un file di Help.
Innanzi tutto bisogna impostare la proprietà *HelpFile*, questo può essere fatto attraverso la finestra "proprietà di progetto" oppure utilizzando un'istruzione simile alla seguente:

```
App.Helpfile="c:\miopercorso\miohelp.chm"
```

Questa impostazione (per le guide HTML dato che il file è un .chm) è sufficiente per attivare la guida quando si preme il tasto *F1*. Per rendere la guida sensibile al contesto, oltre a definire opportunamente le informazioni nel progetto della guida, bisogna impostare la proprietà *HelpContextID* degli oggetti della form.
La *HelpContextID*, attraverso un numero intero, consente di collegare un elemento dell'interfaccia, per esempio una *SSTab*, ad un Topic della guida (identificato da quel numero) così da estrarre informazioni mirate dalla documentazione.
Infine, per rendere la guida rapida sulla form bisogna impostare le proprietà che consentono di attivare il pulsante "punto interrogativo". Questo pulsante, infatti, consente di attivare i messaggi Popup associati ai singoli elementi dell'interfaccia. Come è noto i Popup sono mostrati senza la necessità di visualizzare il file complessivo della guida.
Queste caratteristiche s'impostano attraverso la proprietà *WhatsThisHelp*.
In realtà per attivare il punto interrogativo oltre alla WhatsThisHelp bisogna impostare altre proprietà.
Complessivamente sulla finestra delle proprietà della form bisogna impostare la seguente condizione:

```
WhatsThisButton =True,
BorderStyle =3 - Fixed Dialog,
MaxButton= False,
MinButton= False.
```

Attenzione! queste proprietà sono modificabili solo in fase di progettazione.
Ora siamo pronti per definire i files del progetto WinHelp che successivamente convertiremo in un progetto HTML.

IL PROGETTO WINHELP

Come accennato, in un progetto Winhelp i Topics sono specificati in un file RTF (che possiamo creare per esempio con WinWord). Il file RTF deve essere strutturato secondo le seguenti regole (specifichiamo solo le regole principali). I Topics devono essere delimitati da Interruzione di Pagina. Del Topic devono essere specificati almeno: il nome, il titolo e delle parole chiave. Questi attributi devono essere specificati come note di fine pagina. Le note devono essere scritte rispettando la seguente sintassi:

- una nota con il carattere **#** (non con un numero come è impostato di default) specifica il nome della pagina (possibilmente con caratteri maiuscoli);
- una nota con carattere **$** specifica il Titolo della pagina;
- una nota con carattere **K** specifica le parole chiave (necessarie per la ricerca del Topic).

Poi ci sono altre regole, che non descriviamo, per definire i Link tra Topics e con i POPUP.
Ecco un esempio di file RTF, naturalmente le note vanno inserite a fine pagina attraverso il menu di Word "inserisci / note a piè di pagina".

```
$ # k Prima Pagina
Contenuto prima pagina
….le note vanno inserite a Piè di pagina

 $ Titolo prima pagina
 # PRIMAPAGINA
 K prima
```

Questa struttura deve essere ripetuta per tutti i Topics (nel nostro caso per tre pagine).
Per quanto riguarda il file context (ricordiamo che si tratta di un file del blocco note salvato con estensione .CNT) deve avere la seguente struttura.

```
:Base ESEMPIO.hlp>main
:Title Esempio
:Index Esempio=esempio.hlp
1 informazioni
2 PAGINE
3 INFORMAZIONE PRIMA PAGINA=PRIMAPAGINA
3 INFORMAZIONE SECONDA PAGINA=SECONDAPAGINA
3 INFORMAZIONE TERZA PAGINA=TERZAPAGINA
```

Notate l'organizzazione ad albero con i punti 1, 2 e 3, inoltre notate il collegamento con il nome delle pagine attraverso "…=PRIMAPAGINA". Controllate che cosa produce questo file in Figura.

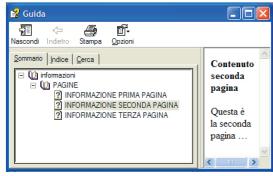



Fig. 2: **La guida HTML di esempio.**

Infine descriviamo il file di progetto (cioè un file formato Ascii salvato con estensione HLP). Esso è diviso in alcune sezioni (è simile a i file .INI di Windows). In particolare può avere le seguenti sezioni:

- **OPTIONS** è la sezione che contiene informazioni per configurare il tipo di compilazione e per specificare alcuni file di supporto (per esempio file di log, file di Context, ecc.);
- **FILES** sezione che specifica i files RTF;
- **MAP** questa sezione serve per assegnare ad ogni Topic un numero di context. Questo numero verrà usato in Visual Basic per impostare la proprietà *HelpContextID*.

Di seguito riportiamo il nostro file di Progetto.

```
[OPTIONS]
HCW=1
REPORT=Yes
HLP=ESEMPIO.hlp
ERRORLOG=ESEMPIO.log
TITLE=QUESTO E' UN UN ESEMPIO
CNT=ESEMPIO.CNT

[FILES]
ESEMPIO.rtf

[MAP]
PRIMAPAGINA 1
SECONDAPAGINA 2
TERZAPAGINA 3
```

I file *ESEMPIO.HPJ* (file progetto), *ESEMPIO.RTF* (file topics), *ESEMPIO.CNT* (file context) devono trovarsi nella stessa directory del compilatore. Nel file notate la sezione di MAP con il nome dei 3 Topics associati ad un numero.
Il comando MsDos da lanciare, per eseguire la compilazione, è simile al seguente.

Visual Basic

Come distribuire l'Help

Per distribuire le guide WinHelp, con un progetto, in generale c'è bisogno soltanto del file .hlp, dato che tutti i sistemi Windows sono forniti del Windows Help Viewer. Il file *.hlp* viene inserito nel pacchetto di installazione del progetto dal Wizard: Creazione guidata pacchetti di installazione.
Anche per la distribuzione delle guide HTML non ci sono particolari problemi dato che tutti i sistemi operativi Microsoft sono equipaggiati con Internet Explorer.

Visual Basic

```
HCP.EXE c:\miopercorso\ESEMPIO.HPJ
```

Questo comando produrrà due file:

- *ESEMPIO.log* che è un file che fornisce informazioni sulla compilazione,
- *ESEMPIO.hlp* che è il file da collegare a Visual Basic.

Noi, però, al progetto Visual Basic collegheremo l'HTML Help che come accennato ricaveremo attraverso il Wizard di conversione del Workshop. In ogni caso, quindi, i files descritti, per il WinHelp, bisogna crearli perchè serviranno al Wizard.

IL PROGETTO HTML HELP

Il Workshop HTML Help (utilizzato per i nostri esempi) lo potete scaricare da *www.microsoft.com/DownLoad*. Sullo stesso sito potete trovare: la guida alla creazione degli HTMLHelp (*HtmlHelp.chm*), l'HTML Viewer (*Viewhlp.chm*), *Hhupd.exe* ecc.
Quest'ultimo file è utile per coloro che non usano Windows XP o Windows 2000 (controllate la documentazione sul sito della Microsoft).
Dopo le installazioni avviate il Workshop e create un nuovo progetto con il Wizard "New Project". Il Wizard si attiva attraverso la voce di menu "File/New … New Project".
Ricordate che sulla prima maschera del Wizard dovete selezionare "Convert WinHelp Project" e poi nei successivi passaggi scegliere la directory dove avete salvato i file del progetto WinHelp.
Il Wizard crea: il file di progetto, il file di contesto e i Topics HTML. Questi ultimi sono inseriti in una directory denominata HTML. Facciamo notare che per quanto riguarda la sezione MAP il Workshop non riesce a convertirla, per questo il Mapping dovete crearlo nuovamente attraverso i menu del Workshop (lo stesso problema si ha con i Popup).

MAPPING

Ora descriviamo come fare il mapping del contesto con il Workshop.
Innanzi tutto bisogna definire un file Ascii con estensione ".h" nel quale inserire le seguenti informazioni:

```
#DEFINE PRIMAPAGINA 1
#DEFINE SECONDAPAGINA 2
#DEFINE TERZAPAGINA 3
```

Con le definizioni precedenti ai nomi dei Topics si associano dei numeri.
Salvate questo file, nominandolo *Esempio.h*, nella stessa directory del file HHP.
Il file *Esempio.h* ora bisogna caricarlo nel progetto *HTMLHelp* attraverso la maschera "HTML Help API Information" che si seleziona con il pulsante "HTML Help API Information" della toolbar verticale della scheda Project. Sulla stessa maschera nella scheda Alias bisogna specificare le seguenti informazioni:

```
PRIMAPAGINA=HTML\esem6au9.htm
SECONDAPAGINA= HTML\esem2sip.htm
TERZAPAGINA= HTML\esem1ild.hml
```

Con le informazioni precedenti si associano i nomi dei Topics alle corrispondenti pagine HTML create dal Wizard (i nomi delle pagine sono state assegnate dal Wizard e corrispondono alle 3 pagine create con il file RTF).
Le impostazioni precedenti servono a rendere la sintassi del Workshop (che è *C Like*) compatibile con la sintassi Visual Basic.
Dopo le operazioni precedenti bisogna compilare il progetto (selezionando l'icona evidenziata in Fig. 3).
Dopo la compilazione, notate che nel file HLP non viene creata la scheda ricerca per crearla bisogna impostare il parametro del progetto *Full Text Search* della sezione *Options*.

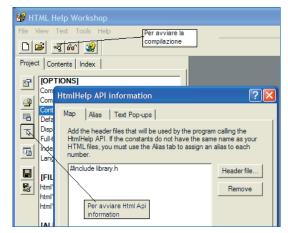



Fig. 3: Il Workshop con la maschera HTMLHelp API Information.

Ora possiamo descrivere il codice che permette di collegare la guida HTML al progetto Visual Basic. Innanzi tutto, vediamo gli elementi dell'API per richiamare la guida da Visual Basic. In particolare introdurremo la funzione HTMLHelp (dell'API HTMLHelp) ed alcune costanti di supporto.
Se invece nel nostro progetto volessimo inserire i Popup dovremmo utilizzare anche la funzione *HtmlHelpByRefArg* ed una serie di oggetti di supporto.

API HTML HELP E CODICE VISUAL BASIC

Allora nel modulo di supporto dobbiamo inserire le seguenti dichiarazioni.

```
Public VarHelp As String
'variabile che contiene il percorso del file Help
```

Office Assistant

Microsoft Office e Windows XP alla guida in linea possono associare anche dei personaggi animati. Per esempio Windows XP permette di scegliere tra i seguenti personaggi: Lalla, Spido, Briciola, Merlino. Il personaggio permette di eseguire delle ricerche nella documentazione e fornisce le informazioni in dei fumetti, questo rende il tutto più semplice e accattivante.
I personaggi animati possono essere usati anche nei progetti Visual Basic attraverso una parte del modello ad oggetti di Office.

```
Const HH_DISPLAY_TOPIC = &H0
`costante necessaria per mostrare la guida
Const HH_HELP_CONTEXT = &HF
 `costante necessaria per selezionare i Topics tramite
                                        HelpContextID
Declare Function HTMLHelp Lib "hhctrl.ocx" _
    Alias "HtmlHelpA" (ByVal hwnd As Long, _
    ByVal lpHelpFile As String, _
    ByVal wCommand As Long, _
    ByVal dwData As Long) As Long
```

Questa funzione avvia l'HTML Help e gli passa alcuni parametri che indicano la natura della richiesta. In particolare:

- **hwnd** è "l'identificativo" della finestra che richiama l'Help;
- **lpHelpFile** è il file di help da richiamare;
- **wCommand** è una costante che specifica come mostrare l'Help;
- **dwdata** è un parametro che serve ad individuare il controllo che richiama l'Help (è collegato alla proprietà HelpContextID).

Per i nostri scopi definiamo anche le seguenti procedure.

```
Public Sub hhdisplay(hwnd As Long, myhelp As String)
  Dim hwndhelp As Long
  hwndhelp = HTMLHelp(hwnd, myhelp, hh_display_topic, 0)
 End Sub
Public Sub hhhelp(hwnd As Long, myhelp As String,
                                   contextid As Long)
 Dim hwndhelp As Long
 hwndhelp = HTMLHelp(hwnd, myhelp,
                        HH_HELP_CONTEXT, contextid)
End Sub
```

La *hhdisplay* serve a mostrare l'Help quando è premuto *F1* invece la *hhhelp* serve a mostrare un particolare argomento dell'Help.
Ora descriviamo il codice da inserire nella form iniziando dalla *Form_Load*.

```
Private Sub Form_Load()
 VarHelp = App.Path & "\htmlhelp\esempio.chm"
 SSTab1.HelpContextID = 1
 Form1.KeyPreview = True
End Sub
```

Con le istruzioni precedenti: impostiamo il percorso dell'Help, fissiamo il *ContextID* della *SSTab* e impostiamo la proprietà *KeyPrevier*. Questa proprietà, impostata su *True*, stabilisce che gli eventi della tastiera per la form sono generati prima di quelli del Controllo attivo (questo per controllare la pressione del tasto *F1*). Infine nella *SSTab1_KeyUp*, per selezionare le informazioni di contesto, prevediamo il seguente codice.

```
Private Sub SSTab1_KeyUp(KeyCode As Integer, Shift
                                       As Integer)
If KeyCode = vbKeyF1 And Shift = 0 Then
  Select Case SSTab1.Tab + 1
   Case 1
    hhhelp Me.hwnd, VarHelp, 1
   Case 2
    hhhelp Me.hwnd, VarHelp, 2
   Case 3
    hhhelp Me.hwnd, VarHelp, 3
  End Select
 End If
End Sub
```

Fig. 4: **Nella figura si vede il Popup associato al Textbox Nome.**

Nella *SSTab1_KeyUp* controlliamo che sia stato premuto il tasto *F1* e poi, in base alla scheda della *SSTab* selezionata, richiamiamo il Topic dell'Help. A tal proposito dovete ricordare che nel progetto dell'Help ad ogni pagina abbiamo attribuito un nome e un numero (con *Esempio.h*).

Fig. 5: **Le icone dei due tipi di Help in Window XP.**

CONCLUSIONE

In questo appuntamento abbiamo illustrato le principali caratteristiche di una guida in linea. Nello spazio di un articolo, però, non abbiamo potuto esaurire l'argomento; infatti non abbiamo illustrato come gestire i Popup, come attivare gli Assistenti animati di Office e come gestire gli errori con la proprietà Help dell'oggetto Err ecc. Questi ed altri argomenti li tratteremo nel successivo articolo, inoltre nel CD della prossima rivista troverete l'applicazione di esempio completa.

Massimo Autiero

Visual Basic

Topics

I Topics devono essere delimitati da Interruzione di Pagina. Del Topic devono essere specificati almeno: il nome, il titolo e delle parole chiave. Questi attributi devono essere specificati come note di fine pagina.

Oggetti

GUIDA ALL'USO

Analizziamo l'ultimo componente di base della programmazione orientata agli oggetti: la variabile oggetto

Nel numero precedente ci siamo occupati del corretto disegno di una classe, ed abbiamo visto come una classe, di per sé, è un pezzo di codice che definisce un concetto astratto. Una classe può essere paragonata a dei modelli che definiscono le caratteristiche di ciascun oggetto. Per utilizzare il codice scritto in una classe si deve creare un oggetto basato sulla classe di riferimento.
La creazione di un oggetto viene detta creazione di un'istanza, ed in VB.Net può essere trattato come una vera e propria variabile. Una volta creati gli oggetti, ognuno avrà i propri valori e potrà eseguire i propri metodi.

LA VARIABILE OGGETTO

Tipicamente per un uso corretto di una variabile oggetto, si distinguono le seguenti fasi:

- Dichiarazione della variabile oggetto
- Creazione dell'oggetto
- Usare le proprietà ed i metodi dell'oggetto
- Rilasciare i riferimenti all'oggetto

Una variabile oggetto può essere dichiarata in qualunque parte del codice con la stessa sintassi di una normale variabile.
A differenza delle variabili normali, che possono essere utilizzate non appena vengono dichiarate, per utilizzare una variabile oggetto si deve creare esplicitamente il riferimento all'oggetto stesso. Se tentiamo di accedere ad una proprietà di un oggetto che non sia stato creato esplicitamente, il compilatore VB genera un'eccezione "Riferimento a un oggetto non impostato su un'istanza di oggetto". Dopo aver creato l'oggetto, con la sintassi che vedremo in seguito, sarà possibile utilizzare una qualsiasi proprietà, provocando l'esecuzione delle procedure *Property* definite nella classe, ed un qualsiasi metodo provocando l'esecuzione delle relative procedure nella classe.
Dopo aver utilizzato la variabile oggetto è buona norma distruggerla esplicitamente, anche se in VB.Net è implementato un meccanismo che si occupa del rilascio automatico delle risorse.
Analizziamo in dettaglio i singoli passi

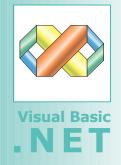
DICHIARAZIONE DI UNA VARIABILE OGGETTO

Una variabile oggetto può essere utilizzata allo stesso modo di una variabile normale (utilizzando ad esempio le istruzioni *Dim*, *Private* o *Public*) e presenta le stesse caratteristiche di visibilità (*scope*) e di durata (*lifetime*).
Così come le normali variabili, VB consente di dichiarare la visibilità delle variabili oggetto a livello di:

- Blocco
- Procedura
- Modulo
- Progetto

Le caratteristiche specifiche di una variabile oggetto sono:

- Possibile utilizzo della parola chiave facoltativa *New* nella dichiarazione della variabile
- L'argomento di tipo classe.

Riprendiamo l'esempio del precedente articolo in cui abbiamo scritto la classe cliente che definisce un generico cliente di un supermercato con le proprietà: *Nome*, *Cognome* e *SpesaMensile;* ed il metodo *CalcolaSpesaAnnua*.

File sul Web

www.itportal.it\iop68\VbNetOOP.zip

Durata di un oggetto

La durata di un oggetto è delimitata dalla creazione e dalla distruzione dell'istanza dell'oggetto stesso. Mentre il momento in cui un oggetto viene creato è ben definibile e determinato dal programma che dichiara l'oggetto, non è attuabile determinare il momento in cui avviene la distruzione. Il Garbage Collector analizza la struttura di riferimenti in background. Se viene rilevato un oggetto o un gruppo di oggetti cui non è associato alcun riferimento presente nel codice correntemente in esecuzione, ne viene chiamato il distruttore. Non è possibile prevedere l'ordine della distruzione o il tempo richiesto per analizzare la struttura di riferimenti. La durata di un oggetto è quindi indeterminata.

Visual Basic
.NET

With... End With

L'istruzione *With...End With* permette di semplificare la scrittura del codice poiché consente di eseguire una serie di istruzioni su un oggetto specificato senza riscrivere il nome dell'oggetto. Se, ad esempio, si vuole modificare una serie di proprietà diverse per un singolo oggetto, risulta più agevole inserire le istruzioni all'interno dell'istruzione *With... End With* facendo riferimento all'oggetto una sola volta:

```
With ObjCliente
    .Nome="Luigi"
    .Cognome="Buono"
    .SpesaMensile=100
End With
```

piuttosto che dover riscrivere il nome della variabile oggetto per ogni assegnazione di proprietà:

```
ObjCliente.Nome="Luigi"
ObjCliente.Cognome=
                "Buono"
ObjCliente.SpesaMensile=
                    100
```

Per dichiarare di un oggetto di tipo *Cliente*, si dovrà scrivere:

```
Dim ObjCliente As cliente
```

La variabile oggetto *ObjCliente* rappresenterà un riferimento ad un oggetto della classe *Cliente.* Se a questo punto si tenta di accedere ad una proprietà dell'oggetto, si solleva un'eccezione poiché la variabile oggetto non è stata ancora creata.
Nella fase di dichiarazione è possibile utilizzare la parola chiave *New,* in questo modo la variabile oggetto sarà creata immediatamente.

```
Dim ObjCliente As New cliente()
```

CREAZIONE DELL'OGGETTO

La creazione di un oggetto può avvenire ovunque nell'area di visibilità della variabile oggetto, purché avvenga prima che l'oggetto sia usato.
Per creare nuovi oggetti, con le caratteristiche definite nella classe corrispondente, si deve utilizzare la parola chiave *New* con la seguente sintassi: (si può notare come non è più necessario l'uso dell'istruzione *Set,* obbligatoria in VB6)

```
ObjCliente = New cliente()
```

Quando la linea di codice viene eseguita, viene creato un oggetto *ObjCliente* istanza della classe cliente e viene eseguito il metodo costruttore che analizzeremo in seguito.

USARE LE PROPRIETÀ ED I METODI DELL'OGGETTO

Una volta creato l'oggetto, le proprietà dichiarate come pubbliche nella classe, possono essere lette e/o modificate.
Le proprietà dotate di una procedura *Property* con la sola clausola *Get* sono a sola lettura, mentre le proprietà dotate di una procedura *Property* con le clausole *Get* e *Set* possono essere lette e modificate. Per accedere alle proprietà di un oggetto si deve usare il punto *(.)* con la seguente sintassi:

```
NomeOggetto.NomeProprietà
```

VB visualizza automaticamente, dopo il punto, un elenco da cui poter selezionare tutte le proprietà, metodi ed eventi utilizzabili dell'oggetto. Per modificare il valore della proprietà *Nome* dell'oggetto *ObjCliente* si dovrà quindi scrivere:

```
ObjCliente.Nome = "Luigi"
```

I metodi definiscono le azioni che possono essere eseguite dall'oggetto. Ogni metodo definito come pubblico può essere eseguito in qualsiasi parte del codice per un determinato oggetto. I metodi definiti privati possono essere utilizzati soltanto all'interno della classe in cui sono stati definiti. Per accedere ai metodi si deve utilizzare, allo stesso modo delle proprietà, il punto dopo la variabile oggetto:

```
SpesaAnnua = ObjCliente.CalcolaSpesaAnnua
```

RILASCIARE I RIFERIMENTI ALL'OGGETTO

Quando la variabile oggetto non deve più essere utilizzata nel codice, deve essere dissociata dall'oggetto, per liberare le risorse di sistema occupate. In VB.Net una variabile oggetto viene distrutta automaticamente quando esce dall'area di visibilità, ma è comunque buona norma distruggerla implicitamente impostandola a *Nothing*:

```
ObjCliente = Nothing
```

Nelle precedenti versioni di VB l'uso della parola chiave *Nothing* era molto importante ed obbligatoria. In VB.Net, con l'implementazione del meccanismo di *Garbage Collection*, non viene più garantito il rilascio immediato degli oggetti. La documentazione consiglia di assegnare a *Nothing* solo oggetti con durata estesa, come membri condivisi o variabili globali

CARATTERISTICHE DELLE VARIABILI OGGETTO

A questo punto è necessario fissare alcuni concetti sull'uso delle variabili oggetto in modo da evitare errori difficili da rintracciare. Una variabile oggetto non è un area di memoria che contiene i dati dell'oggetto, ma è piuttosto un puntatore ad un area di memoria in cui sono memorizzati i dati dell'oggetto.
Questo concetto deve essere sempre presente nella fase di programmazione poiché ne conseguono alcune caratteristiche peculiari molto importanti:

- Ogni variabile oggetto essendo un puntatore ad un area di memoria occuperà sempre lo stesso spazio indipendentemente dalle dimensioni dell'oggetto cui fa riferimento (in VB quattro byte)

- Ogni volta che si pone una variabile oggetto

uguale ad un'altra, non verrà duplicato nessun dato ma in realtà verrà assegnato alla nuova variabile oggetto lo stesso indirizzo in memoria della precedente.

- Se due o più variabili oggetto indicano la stessa istanza dell'oggetto, esisterà soltanto una copia delle proprietà. Per questo, modificando il valore di una proprietà di una qualsiasi di queste variabili, tutte le altre variabili vedranno immediatamente il nuovo valore.

Vediamo un esempio delle conseguenze delle ultime due caratteristiche. Supponiamo di avere due oggetti differenti di tipo cliente: *ObjCliente* e *ObjCliente2*. Dopo aver creato l'oggetto *ObjCliente*, settiamo la proprietà *Nome* pari a *"Luigi"* e la proprietà *Cognome* pari a *"Buono"*; e visualizziamole in una finestra *MessageBox*, con il seguente codice:

```
Dim ObjCliente As cliente
Dim ObjCliente2 As cliente
ObjCliente = New cliente()
ObjCliente.Nome = "Luigi"
ObjCliente.Cognome = "Buono"
ObjCliente.SpesaMensile = "1000"
MessageBox.Show(ObjCliente.Cognome + " "
                            + ObjCliente.Nome)
```

Il risultato sarà, banalmente, una finestra che mostra la stringa *"Buono Luigi"*. A questo punto supponiamo di avere un secondo cliente con le stesse caratteristiche del primo ma che abbia il nome differente.

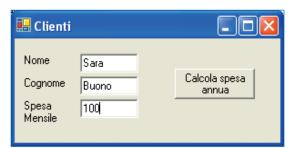
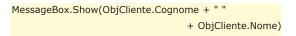



Fig. 1: Finestra "Message Box".

Per risparmiare istruzioni ci potrebbe venire in mente di porre il secondo oggetto *ObjCliente2* uguale ad *ObjCliente* e settare la proprietà differente *Nome="Sara"*

```
ObjCliente2 = ObjCliente
ObjCliente2.Nome = "Sara"
```

Se ora visualizziamo di nuovo le proprietà del primo oggetto con la stessa istruzione usata in precedenza:

```
MessageBox.Show(ObjCliente.Cognome + " "
                            + ObjCliente.Nome)
```

Cosa vi aspetterete di veder comparire a video? Ebbene comparirà la stringa *"Buono Sara"* a conferma di ciò che è stato affermato in precedenza.

COSTRUTTORI

Un aspetto molto importante della progettazione di oggetti è il concetto di costruttore. Il costruttore è una porzione di codice di inizializzazione eseguito ogni volta che viene creata un'istanza di un oggetto. Tipiche operazioni d'inizializzazione sono: l'apertura di file, la connessione ad un database, o più semplicemente l'impostazione di valori predefiniti per le proprietà di un oggetto.
Per creare un costruttore, si deve scrivere una procedura denominata *Sub New* in un qualsiasi punto del codice della classe. Ad esempio nella classe cliente si può scrivere la seguente procedura per inizializzare il valore di spesa mensile ad un valore forfetario di 50 Euro

```
Public Sub New()
    SpesaMensile = 50
End Sub
```

Nella definizione di un costruttore può essere necessaria la richiesta di parametri. In questo caso si possono definire i nomi ed i tipi di dati degli argomenti in modo analogo alla definizione degli argomenti per qualsiasi altra procedura. Ad esempio nella classe cliente si può scrivere il seguente costruttore che chiede di passare come parametro il valore della spesa mensile.

```
Public Sub New(ByVal Spesa As Decimal)
    SpesaMensile = spesa
End Sub
```

I due costruttori dell'esempio possono coesistere nella classe cliente per la caratteristica dell'*overloading*. L'*overloading*, introdotto nell'articolo precedente, consiste nella possibilità di definire più versioni di uno stesso metodo all'interno della stessa classe, utilizzando lo stesso nome ma un numero e/o un tipo diverso di argomenti. In fase di esecuzione, a seconda se passiamo o no il valore di spesa nella procedura, verrà richiamato il costruttore corretto.
Il codice del metodo *Sub New* viene eseguito prima del resto del codice della classe. Se non si definisce in modo esplicito una *Sub New*, VB .NET crea in modo implicito, in fase di esecuzione, un costruttore *Sub New*. Le routine *Sub New* sostituiscono i metodi *Class_Initialize* presenti in VB6.

Visual Basic .NET

Variabili Object

Le variabili di tipo Object possono contenere un riferimento ad un qualsiasi oggetto e vengono memorizzate sempre come indirizzi di 32 bit (4 byte). L'associazione all'oggetto cui fa riferimento una variabile di tipo Object è sempre in fase di esecuzione (associazione tardiva) per questo l'utilizzo della variabili di tipo Object rallenta le prestazioni del sistema. Per ottenere l'associazione in fase di compilazione (associazione anticipata), certamente più efficiente, si deve assegnare ad una variabile il riferimento all'oggetto con un nome di classe specifico.

Visual Basic .NET

TypeOf

Per determinare se una variabile oggetto fa riferimento a un tipo di dati specifico è possibile utilizzare l'operatore TypeOf. L'espressione TypeOf...Is restituisce True se il tipo in fase di esecuzione dell'oggetto implementa il tipo specificato. es

```
Dim ObjGenerico As
        Object = 10
   ' ...
If TypeOf ObjGenerico
        Is Long Then
   MessageBox.Show(
     "ObjGenerico è un
               long")
End If
If TypeOf ObjGenerico
        Is Integer Then
   MessageBox.Show(
     "ObjGenerico è un
            integer")
End If
If TypeOf ObjGenerico
        Is cliente Then
   MessageBox.Show(
     "ObjGenerico è un
            cliente")
End If
```

Ragionare per oggetti è "naturale"... perché:

- **Facilmente osservabili nel mondo reale**
- **Definibili per astrazione**
- **Vicini all'utente**
- **Facilmente organizzabili in gerarchie**
- **Riusabili**

DISTRUTTORI

Il distruttore è una porzione di codice per il rilascio delle risorse di sistema. Viene eseguito ogni volta che si rilascia il riferimento ad un oggetto. Tipiche operazioni di rilascio delle risorse di sistema sono: la chiusura di un file, la chiusura di una connessione a database o il salvataggio d'informazioni di stato. Prima di distruggere un oggetto, il compilatore chiama automaticamente il metodo *Finalize* per oggetti che definiscono una routine *Sub Finalize* (in VB6 era *Class_Terminate*). Per il meccanismo di Garbage Collection implementato in VB.Net è possibile che si verifichi un ritardo dal momento in cui un oggetto viene impostato su *Nothing*, e la chiamata del distruttore *Finalize*, per questo si può chiamare in modo esplicito un secondo tipo di distruttore, denominato *Dispose*, in qualunque momento per rilasciare immediatamente le risorse.

IL MECCANISMO DI GARBAGE COLLECTION

Nelle precedenti versioni di VB veniva utilizzato un meccanismo, definito conteggio dei riferimenti, per la gestione delle risorse impegnate dalle variabili oggetto.
In pratica ogni istanza di oggetto tiene un conteggio dei riferimenti. Quando l'ultimo riferimento a un'istanza viene rilasciato e il conteggio viene azzerato, l'oggetto cessa di esistere immediatamente. In VB.NET viene utilizzato, invece, un sistema definito Garbage Collection ad analisi dei riferimenti, che controlla in modo implicito tutte le attività di gestione della memoria e prevede il rilascio periodico delle risorse non utilizzate.
Mediante questo metodo gli oggetti vengono eliminati periodicamente quando il sistema stabilisce che non sono più necessari. La frequenza con cui il compilatore rilascia gli oggetti non utilizzati, aumenta quando le risorse di sistema sono insufficienti e diminuisce in caso contrario. Questo rende impossibile determinare il momento esatto in cui viene lanciato il distruttore *Finalize.* Nella maggior parte dei casi, questo comportamento non influisce sulla modalità di scrittura delle applicazioni, purché ci si ricordi della possibilità che il distruttore *Finalize* non venga eseguito istantaneamente dopo la perdita di ambito di un oggetto.

APPLICAZIONE DI ESEMPIO

Per meglio fissare i concetti espressi finora realizziamo una semplice applicazione di esempio sull'uso degli oggetti.
Sfruttiamo la classe cliente scritta nel presente articolo e realizziamo una form contenente tre TextBox: *TextBoxNome*, *TextBoxCognome* e *TextBoxSpesaMensile* con tre Label corrispondenti ed un Button: *ButtonCalcolaSpesa*. Nei tre TextBox scriveremo il *Nome*, *Cognome* e *Spesa Mensile* di un generico cliente di un supermercato. Premendo il pulsante, verrà mostrato un messaggio con il riassunto dei dati del cliente inserito, ed una stima della sua spesa annuale, calcolata grazie al metodo *CalcolaSpesaAnnua* della classe cliente:

```
Public Function CalcolaSpesaAnnua() As Decimal
    CalcolaSpesaAnnua = SpesaMensile * 12
End Function
```

Per questo il codice nell'evento *Click* di *ButtonCalcolaSpesa* sarà:

```
Private Sub ButtonCalcolaSpesa_Click(ByVal sender
        As System.Object, ByVal e As System.EventArgs)
                        Handles ButtonCalcolaSpesa.Click
    'dichiarazione della variabile oggetto di tipo cliente
    Dim ObjCliente As cliente
    'creazione di un nuovo oggetto ObjCliente
    ObjCliente = New cliente()
    'assegnazione dei valori alle proprietà di ObjCliente
    ObjCliente.Nome = TextBoxNome.Text
    ObjCliente.Cognome = TextBoxCognome.Text
    ObjCliente.SpesaMensile = CDec(
                        TextBoxSpesaMensile.Text)
    'visualizzazione a video di un messaggio in cui
         vengono indicati i dati del cliente con la sua
                        spesa annuale calcolata
    MessageBox.Show(ObjCliente.Nome & " "
                        & ObjCliente.Cognome & _
              " spende annualmente euro: " & CStr(
                        ObjCliente.CalcolaSpesaAnnua))
    'rilascio dei riferimenti all'oggetto
    ObjCliente = Nothing
End Sub
```

Nel codice sono state commentate tutte le singole istruzioni per questo tralasciamo la descrizione dettagliata di ogni riga. Potete notare le singole fasi di: dichiarazione, creazione, utilizzo di proprietà e metodi dell'oggetto e distruzione della variabile oggetto, descritte nell'introduzione dell'articolo

CONCLUSIONI

In questo articolo abbiamo descritto l'ultimo tassello di base che ci mancava nella descrizione della OOP, nel prossimo articolo introdurremo il concetto di collezione di oggetti aggiungendo un'ulteriore tessera nel mosaico della programmazione ad oggetti.

Ing. Luigi Buono

Sovraccarico
DEGLI OPERATORI (PARTE SECONDA)

C#

Sovraccaricare gli operatori è una tecnica spesso utile, quando si elaborano nuovi tipi di dati.

Nel corso della lezione precedente abbiamo approcciato il sovraccarico degli operatori. Abbiamo imparato come sovraccaricare gli operatori aritmetici, sia binari sia unari. Oggi andremo avanti con lo studio intrapreso, esaminando il sovraccarico degli operatori di confronto.

GLI OPERATORI DI CONFRONTO

C# offre sei operatori di confronto:

Operatore	Significato
==	Uguale
!=	Diverso
>	Maggiore
>=	Maggiore o uguale
<	Minore
<=	Minore o uguale

Questi operatori si sovraccaricano sempre a coppie di due:

- == insieme a !=
- > insieme a <
- >= insieme a <=

Non è possibile sovraccaricare il primo elemento di una coppia senza sovraccaricare anche il suo collega, e viceversa. Gli operatori di confronto, inoltre, devono sempre restituire un risultato di tipo booleano (un bool, per dirlo come desidera C#). Riprendiamo la classe *Vettore2D*, già impiegata nel corso della lezione precedente per dimostrare il sovraccarico degli operatori aritmetici:

```
class Vettore2D {
  public int i;
  public int j;
  public Vettore2D(int ci, int cj) {
    i = ci;
    j = cj; }
}
```

Sovraccarichiamo la prima coppia di operatori di confronto (== e !=). I modelli da impiegare sono i seguenti:

```
public static bool operator ==(tipo1 arg1, tipo2 arg2) {
  // ... }
public static bool operator =!(tipo1 arg1, tipo2 arg2) {
  // ...
}
```

Due vettori sono uguali quando si equivalgono in ambo le loro componenti i e j. Quindi, nel caso di Vettore2D, arriveremo ad un codice come il seguente:

```
class Vettore2D {
  public int i;
  public int j;
  public Vettore2D(int ci, int cj) {
    i = ci;
    j = cj; }
  public static bool operator ==(Vettore2D a, Vettore2D b) {
    return ((a.i == b.i) && (a.j == b.j)); }
  public static bool operator !=(Vettore2D a, Vettore2D b) {
    return !(a == b); }
}
```

Proviamo a compilare il codice. Inaspettatamente, otterremo due errori assai strani:

```
'Vettore2D' defines operator == or operator != but does
                    not override Object.Equals(object o)
'Vettore2D' defines operator == or operator != but does
                    not override Object.GetHashCode()
```

Come mai il tentativo non ha funzionato? Per motivi che potremo comprendere solo più avanti, il sovraccarico degli operatori == e != può essere eseguito solo dopo aver definito i metodi:

```
public override bool Equals(object o)
public override int GetHashCode()
```

Il primo ha una funzione simile a ==, mentre il secondo è impiegato per associare un identificatore numerico ad ogni oggetto istanziato. Giacché ora non siamo in grado di comprendere a pieno lo scopo ed il significato dei due metodi, accontentiamoci di definizioni

File sul CD
\soft\codice\CorsoC#.zip

Agenda

C#, in maniera semplice e flessibile, permette il sovraccarico degli operatori, analogamente a quanto avviene con C++. In questa lezione prenderemo in esame il sovraccarico degli operatori di confronto.

C#

approssimative, ideate al solo scopo di soddisfare i requisiti necessari per il sovraccarico di == e !=:

```
class Vettore2D {
  public int i;
  public int j;
  public Vettore2D(int ci, int cj) {
    i = ci;
    j = cj; }
  public static bool operator ==(Vettore2D a, Vettore2D b) {
    return ((a.i == b.i) && (a.j == b.j)); }
  public static bool operator !=(Vettore2D a, Vettore2D b) {
    return !(a == b); }
  public override bool Equals(object o) {
    return (this == (Vettore2D)o); }
  public override int GetHashCode() {
    return i + j; }
}
```

Bene, a questo punto possiamo eseguire un test, scrivendo da qualche parte un *Main()* contenente il seguente codice:

```
Vettore2D v1 = new Vettore2D(2, 4);
Vettore2D v2 = new Vettore2D(3, 7);
Vettore2D v3 = new Vettore2D(2, 4); // v3 è uguale a v1
System.Console.WriteLine("v1 uguale a v2: " + (v1 == v2));
System.Console.WriteLine("v1 uguale a v3: " + (v1 == v3));
System.Console.WriteLine("v1 diverso da v2: " + (v1 != v2));
System.Console.WriteLine("v1 diverso da v3: " + (v1 != v3));
```

L'output prodotto da questo frammento è il seguente:

v1 uguale a v2: False
v1 uguale a v3: True
v1 diverso da v2: True
v1 diverso da v3: False

Proviamo ora il sovraccarico degli operatori > e <. Prima di eseguire l'operazione sulla classe *Vettore2D*, è necessario stabilire una regola che decreti quando un vettore bidimensionale sia maggiore o minore di un altro. Stabiliamo, per nostra convenzione, che un vettore *a* è maggiore di un vettore *b* quando la misura di *a* è maggiore della misura di *b*. La misura di un vettore, altrimenti detta modulo, si può calcolare in maniera molto semplice:

```
|a| = radice_quadrata(i2 + j2)
```

In parole semplici, la misura di un vettore bidimensionale si ottiene con una semplice applicazione del teorema di pitagora, dove le componenti *i* e *j* del vettore rappresentano i cateti di un triangolo rettangolo avente la rappresentazione grafica del vettore come ipotenusa. Chiarito il fatto, usiamo i seguenti modelli per il sovraccarico degli operatori > e <:

```
public static bool operator >(tipo1 arg1, tipo2 arg2) {
  // ... }
public static bool operator <(tipo1 arg1, tipo2 arg2) {
  // ... }
```

Ecco cosa si ottiene:

```
class Vettore2D {
  public int i;
  public int j;
  public Vettore2D(int ci, int cj) {
    i = ci;
    j = cj; }
  public static bool operator ==(Vettore2D a, Vettore2D b) {
    return ((a.i == b.i) && (a.j == b.j)); }
  public static bool operator !=(Vettore2D a, Vettore2D b) {
    return !(a == b); }
  public override bool Equals(object o) {
    return (this == (Vettore2D)o); }
  public override int GetHashCode() {
    return i + j; }
  public double CalcolaMisura() {
    return System.Math.Sqrt((i * i) + (j * j)); }
  public static bool operator >(Vettore2D a, Vettore2D b) {
    return (a.CalcolaMisura() > b.CalcolaMisura()); }
  public static bool operator <(Vettore2D a, Vettore2D b) {
    return (a.CalcolaMisura() < b.CalcolaMisura()); }
}
```

Per prima cosa, è stato definito un metodo ausiliaro, chiamato *CalcolaMisura()*. Il suo compito è calcolare la misura del vettore corrente. È conveniente definire tale metodo, poiché il calcolo della misura sarà successivamente impiegato in più casi. Bene, compilata la classe, è possibile effettuare un test dimostrativo:

```
Vettore2D v1 = new Vettore2D(2, 4);
Vettore2D v2 = new Vettore2D(3, 7);
Vettore2D v3 = new Vettore2D(1, 2);
System.Console.WriteLine("v1 maggiore di v2: " +
                                        (v1 > v2));
System.Console.WriteLine("v1 maggiore di v3: " +
                                        (v1 > v3));
System.Console.WriteLine("v1 minore di v2: " +
                                        (v1 < v2));
System.Console.WriteLine("v1 minore di v3: " +
                                        (v1 < v3));
```

L'output è, come ci aspettiamo, il seguente:

v1 maggiore di v2: False
v1 maggiore di v3: True
v1 minore di v2: True
v1 minore di v3: False

Tutto questo è corretto, poiché:

- La misura di *v1* è circa 4.47.
- La misura di *v2* è circa 7.61.
- La misura di *v3* è circa 2.23.

Tipi personalizzati

Il sovraccarico degli operatori è uno dei principali pregi di C#. Benché le tecniche adottate da C# non siano complete ed estese come quelle scelte da C++, le possibilità concesse tornano utili quando si progettano delle classi che svolgono la funzione di tipi personalizzati.

Passiamo ora all'implementazione del sovraccarico per gli operatori >= e <=. I modelli sono i seguenti:

```
public static bool operator >=(tipo1 arg1, tipo2 arg2) {
  // ... }
public static bool operator <=(tipo1 arg1, tipo2 arg2) {
  // ... }
```

A questo punto, la scrittura del codice necessario è immediata:

```
class Vettore2D {
  public int i;
  public int j;
  public Vettore2D(int ci, int cj) {
    i = ci;
    j = cj; }
  public static bool operator ==(Vettore2D a, Vettore2D b) {
    return ((a.i == b.i) && (a.j == b.j)); }
  public static bool operator !=(Vettore2D a, Vettore2D b) {
    return !(a == b); }
  public override bool Equals(object o) {
    return (this == (Vettore2D)o); }
  public override int GetHashCode() {
    return i + j; }
  public double CalcolaMisura() {
    return System.Math.Sqrt((i * i) + (j * j)); }
public static bool operator >(Vettore2D a, Vettore2D b) {
    return (a.CalcolaMisura() > b.CalcolaMisura()); }
  public static bool operator <(Vettore2D a, Vettore2D b) {
    return (a.CalcolaMisura() < b.CalcolaMisura()); }
  public static bool operator >=(Vettore2D a, Vettore2D b) {
    return (a.CalcolaMisura() >= b.CalcolaMisura()); }
  public static bool operator <=(Vettore2D a, Vettore2D b) {
    return (a.CalcolaMisura() <= b.CalcolaMisura()); }
}
```

Anche questa volta, per semplificare i confronti, si è fatto ricorso al metodo *CalcolaMisura()* elaborato in precedenza. Con un test come il seguente

```
Vettore2D v1 = new Vettore2D(2, 4);
Vettore2D v2 = new Vettore2D(3, 7);
Vettore2D v3 = new Vettore2D(4, 2);
System.Console.WriteLine("v1 maggiore o uguale di v2:
                                        " + (v1 >= v2));
System.Console.WriteLine("v1 maggiore o uguale v3: "
                                          + (v1 >= v3));
System.Console.WriteLine("v1 minore o uguale v2: " +
                                              (v1 <= v2));
System.Console.WriteLine("v1 minore o uguale v3: " +
                                              (v1 <= v3));
```

si ottiene il risultato

v1 maggiore o uguale di v2: False
v1 maggiore o uguale v3: True
v1 minore o uguale v2: True
v1 minore o uguale v3: True

In effetti, *v1* e *v3* hanno misure equivalenti. Dunque, è logico che il secondo ed il quarto confronto diano esito positivo. Il vettore *v2*, invece, ha misura maggiore di *v1*, per tanto il primo confronto ha esisto negativo, mentre il terzo dà un risultato positivo.

CONFRONTARE TIPI DIFFERENTI

Finora abbiamo sempre messo a confronto due valori del medesimo tipo. Nel caso degli esempi più recenti, in particolare, abbiamo sempre paragonato due oggetti di tipo *Vettore2D*. Nel corso della lezione precedente, ad ogni modo, abbiamo visto come sia possibile definire operazioni tra valori di un tipo e valori di un altro. Lo stesso può essere fatto con i confronti, quando l'operazione ha logicamente senso. Approfittiamo ancora una volta del concetto di misura di un vettore, e andiamo a definire dei confronti tra oggetti *Vettore2D* e valori numerici (*double*, per la precisione).
Ecco come fare:

```
class Vettore2D {
  public int i;
  public int j;
  public Vettore2D(int ci, int cj) {
    i = ci;
    j = cj;
  }
  public static bool operator ==(Vettore2D a, Vettore2D b) {
    return ((a.i == b.i) && (a.j == b.j));
 }
  public static bool operator !=(Vettore2D a, Vettore2D b) {
    return !(a == b); }
  public override bool Equals(object o) {
    return (this == (Vettore2D)o);
  }
  public override int GetHashCode() {
    return i + j; }
  public double CalcolaMisura() {
    return System.Math.Sqrt((i * i) + (j * j)); }
  public static bool operator >(Vettore2D a, Vettore2D b) {
    return (a.CalcolaMisura() > b.CalcolaMisura());
  }
  public static bool operator <(Vettore2D a, Vettore2D b) {
    return (a.CalcolaMisura() < b.CalcolaMisura());
  }
  public static bool operator >=(Vettore2D a, Vettore2D b) {
    ...
    return (v.CalcolaMisura() <= d);
  }
}
```

Stando al codice, un vettore *v* è uguale ad un numerico *d* quando la misura di *v* è uguale a *r*. Analogamente avviene con gli operatori maggiore e minore. Le affermazioni possono essere verificate mediante un test:

```
Vettore2D v = new Vettore2D(4, 3);
```

C#

Sul Web

Tra le dispense Web del corso troverete appunti, aggiunte, approfondimenti, risposte a domande frequenti e altre appendici legate alla lezione odierna. L'indirizzo di riferimento è

www.sauronsoftware.it/dispenseweb/csharp/

C#

```
System.Console.WriteLine("v uguale 5: " + (v == 5));
System.Console.WriteLine("v maggiore 10: " + (v > 10));
System.Console.WriteLine("v minore 11: " + (v < 11));
```

L'output riscontrato è il seguente:
v uguale 5: True
v maggiore 10: False
v minore 11: True

La misura di *v* è *5*, pertanto tutti i confronti effettuati hanno dato l'esito corretto.
Proviamo la compilazione del seguente test:

```
Vettore2D v = new Vettore2D(4, 3);
System.Console.WriteLine("5 uguale v: " + (5 == v));
System.Console.WriteLine("10 maggiore v: " + (10 > v));
System.Console.WriteLine("11 minore v: " + (11 < v));
```

Si ottengono degli errori del tipo:

```
Operator '==' cannot be applied to operands of type
                                    'int' and 'Vettore2D'
Operator '>' cannot be applied to operands of type 'int'
                                          and 'Vettore2D'
Operator '<' cannot be applied to operands of type 'int'
                                          and 'Vettore2D'
```

Questo avviene poiché gli operatori di confronto non sono commutativi. Per risolvere il problema, bisogna definire altri sei operatori di confronto, simmetrici rispetto ai precedenti:

```
class Vettore2D {
  public int i;
  public int j;
  public Vettore2D(int ci, int cj) {
    i = ci;
    j = cj;
  }
  public static bool operator ==(Vettore2D a, Vettore2D b) {
    return ((a.i == b.i) && (a.j == b.j));
  }
  public static bool operator !=(Vettore2D a, Vettore2D b) {
    return !(a == b);
  }
  public override bool Equals(object o) {
    return (this == (Vettore2D)o);
  }
  public override int GetHashCode() {
    return i + j;
  }
  public double CalcolaMisura() {
    return System.Math.Sqrt((i * i) + (j * j));
  }
  public static bool operator >(Vettore2D a, Vettore2D b) {
    return (a.CalcolaMisura() > b.CalcolaMisura());
  }
  ...
}
```

Ora il test tentato in precedenza è divenuto possibile, e l'esito è il seguente:

5 uguale v: True
10 maggiore v: True
11 minore v: False

SOVRACCARICO DI ALTRI OPERATORI

La seguente tabella riporta gli operatori di cui C# consente il sovraccarico:

Categoria	Operatori
Aritmetici binari	+ - * / %
Aritmetici unari	+ - ++ --
Operatori sui bit binari	& | ^ << >>
Operatori sui bit unari	! ~
Confronto	== != > >= < <=

Abbiamo esaminato il sovraccarico di tutti gli operatori aritmetici, sia binari sia unari, e di tutti quelli di confronto. Il sovraccarico degli operatori sui bit è consentito, ma viene effettuato molto raramente, solo quando le cause logiche lo richiedono. Per questo motivo, non esamineremo ora dei casi di sovraccarico degli operatori sui bit.
C#, ad ogni modo, dispone di numerosi altri operatori. Come mai solo una minoranza gode della possibilità di poter essere sovraccaricato? Ci sono tre risposte a questa domanda:

1. Per alcuni operatori il sovraccarico non ha senso. Quindi, è inutile consentirlo.

2. Alcuni operatori vengono sovraccaricati implicitamente. Ad esempio, sovraccaricando l'operatore + si ottiene automaticamente anche il sovraccarico dell'operatore +=.

3. Per gli operatori [] e () il sovraccarico si effettua con tecniche differenti da quelle esaminate nelle più recenti lezioni, attraverso gli indicizzatori ed i casting personalizzati. Esamineremo questi argomenti più in là, quando li incontreremo nel nostro percorso di studio.

I programmatori C++ noteranno, non senza meraviglia, che C# non consente in alcun modo il sovraccarico dell'operatore =. In parole semplici, non è possibile tramutare l'assegnazione in una "duplicazione di oggetti", come di solito avviene in ambito C++.
In C#, l'operatore = compie sempre e solamente il suo compito predefinito: copia i valori quando gestisce l'assegnazioni su variabili di tipo valore, e copia i riferimenti quando tratta con tipi riferimento (come gli oggetti).

Carlo Pelliccia

Bibliografia

• **GUIDA A C#**
Herbert Schildt
(McGraw-Hill)
ISBN 88-386-4264-8
2002

• **INTRODUZIONE A C#**
Eric Gunnerson
(Mondadori Informatica)
ISBN 88-8331-185-X
2001

• **C# GUIDA PER LO SVILUPPATORE**
Simon Robinson e altri
(Hoepli)
ISBN 88-203-2962-X
2001

I template

TECNICHE E VANTAGGI

C++

In questo appuntamento vedremo, come promesso la volta scorsa, un modo per incrementare il riutilizzo di codice nei nostri programmi: l'uso dei template.

Quando si parla di riutilizzo di codice, si intendono principalmente due cose: aumentare l'affidabilità del codice, e lavorare di meno (producendo le stesse cose in minor tempo e scrivendo meno codici). Abbiamo visto come, facendo uso di ereditarietà e classi base astratte, è possibile arrivare a un livello di riutilizzo di codici già alto (oltre che a un modo migliore di modellare le situazioni che ci si presentano davanti); l'uso dei template è un altro modo, che però concettualmente si discosta molto dall'ereditarietà (anche se la combinazione di template ed ereditarietà per un programmatore è, o dovrebbe essere, la normalità). Iniziamo con anticipare una sottile differenza tra il riuso fornito dall'ereditarietà e quello che ci consente l'uso dei template: mediante l'ereditarietà, riusiamo degli oggetti già pronti, mentre con i template, riusiamo codice nel senso letterale del termine (cioè il compilatore va a compilare più volte lo stesso codice, riadattandolo a seconda delle necessità). L'idea che c'è dietro è quella di produrre strutture parametriche, le quali poi vengono usate istanziando di volta in volta i parametri. Non bisogna andare lontano per trovare un analogo esempio di uso di template; se pensiamo al concetto di insieme algebrico, ecco che abbiamo a che fare con un primo template: un insieme è infatti un contenitore di elementi, ed il concetto di insieme lo abbiamo tutti, indipendentemente dal tipo di elementi contenuti nell'insieme. Le operazioni di aggiunta di un elemento all'insieme, di scelta (o eliminazione) di un elemento dall'insieme, il contare (quando possibile) il numero di elementi di un insieme, sono operazioni intuitive che sappiamo almeno pensare senza riferirci al tipo specifico di elementi che costituiscono l'insieme stesso. Ad esempio potremmo pensare ad un insieme di interi, ad un insieme di figure geometriche, ad un insieme di persone, o di nazioni, e così via: queste sono istanze di insiemi. In altre parole, da una parte c'è il concetto di insieme, con le relative operazioni, e dall'altra ci sono gli insiemi istanziati, in cui le operazioni note a livello concettuale sono applicabili direttamente, senza per questo dover pensare ad una estensione del concetto di insieme. Un insieme, e le operazioni definibili su di esso, non cambiano la propria natura quando gli insiemi vengono istanziati: l'eliminazione di un elemento da un insieme è una operazione che si può fare su un insieme, mentre l'eliminazione di un intero da un insieme di interi è esattamente la stessa cosa, ma riferita ad un caso specifico. Ma c'è di più, dietro. Quando si ha bisogno di un insieme di elementi, non si ha bisogno di un tipo specifico di elemento, ma solo delle funzionalità concettuali fornite dalla struttura dati "insieme". A questo punto, dovrebbe iniziare a delinearsi dov'è il problema (e il motivo) che ha spinto a implementare i template: il concetto di insieme può essere reso, infatti, in due modi simili all'atto pratico, ma diversi se visti in prospettiva. Il primo modo, tipico di linguaggi più procedurali (come il C) e che ha come unico aspetto positivo, in generale, le prestazioni migliori, è quello di scrivere codice integrando il concetto di insieme all'interno della logica applicativa. Ad esempio, la funzionalità di aggiunta di un elemento ad un insieme, non può prescindere dal tipo di elementi: se abbiamo bisogno di un insieme di interi, avremo la classe *InsiemeDiInteri*, con la funzione *AggiungiElemento(int)*. Nel momento in cui ci servisse un insieme di persone, la soluzione unica sarebbe: copia e incolla del codice di *InsiemeDiInteri*, sostituzione del nome della classe con *InsiemeDiPersone*, modifica della funzione di aggiunta di un intero all'insieme con *AggiungiElemento(Persona)*, e così di seguito. Stressante, oltre che poco creativo ed entusiasmante. Il secondo modo è caratteristico (ma non originale) del C++: si scrive un template che implementa le funzionalità richieste, usando uno o più parametri che verranno istanziati all'atto pratico. Tradotto, significa che creeremo una classe *Insieme<Elemento>*, intendendo riferirci, con questa notazione, al concetto di *"Insieme di oggetti Elemento"*, e poi decideremo cosa sia un *Elemento* quando se ne presenterà la necessità, istanziando, ad esempio, *Insieme<int>* oppure *Insieme<Scheda>*.

CLASSI E TEMPLATE

L'uso di template in C++ è finalizzato alla creazione di

File sul CD
\soft\codice\cpluscod.zip

Ereditarietà e Template

Mediante l'ereditarietà, riusiamo degli oggetti già pronti, mentre con i template, riusiamo codice nel senso letterale del termine (cioè il compilatore va a compilare più volte lo stesso codice, riadattandolo a seconda delle necessità). L'idea che c'è dietro è quella di produrre strutture parametriche, le quali poi vengono usate istanziando di volta in volta i parametri.

classi il più possibile generiche, di modo da implementare algoritmi che ne facciano uso e ottenendo, transitivamente, l'estensione dell'utilizzabilità degli algoritmi anche a quelle classi modellabili tramite tali classi generiche. In C++ si può progettare una classe parametrica rispetto ad un generico tipo. Ad esempio si può creare, in analogia con quanto visto al termine del paragrafo precedente, una classe *Insieme<T>* la quale, al suo interno, faccia uso del tipo *T* come di un qualsiasi tipo conosciuto: l'effettiva istanza della classe prevedrà anche l'istanza dello specifico tipo *T* del parametro. In altri termini, il percorso è composto di due passi:

1. dichiarazione della classe "insieme di elementi di tipo *T*", in cui invitiamo il compilatore a comportarsi come se *T* lo conoscesse già; gli diciamo "questa è la classe *Insieme*, il tipo *T* fai finta di conoscerlo, tanto saprai, quando necessario, quale sia effettivamente";

2. creazione di una istanza della classe Insieme, in cui esplicitiamo anche quale sia il fantomatico tipo *T* del parametro; nell'ipotesi di un insieme di interi, dobbiamo far capire al compilatore che istanziamo la classe *Insieme* con elementi di tipo *int*.

Utilizzo dei template

Si noti che i template possono venire utilizzati anche in altri modi, non solo in abbinamento a classi.
Per un interessante esempio di uso dei template, vi rimandiamo al codice allegato.

Vediamo la sintassi con cui si definisce una classe template in C++:

```
template<class T>
class C
{
  //funzioni della classe, ad esempio:
  T f(&T);  //si noti l'uso del tipo T
            //come un tipo qualsiasi
};
```

In questo modo, anticipiamo al compilatore che la *classe C* è un template, cioè una classe "stampo" in cui il parametro *T* (che è un tipo) deve essere considerato noto. A questo punto, nella dichiarazione della classe (come si vede dal codice) si può usare il tipo *T* come fosse un tipo qualsiasi: si pensi a *T* come ad int, e si capirà in che senso esso viene inteso dal compilatore. Nel momento della definizione dei metodi della classe template, dovremo ricordare al compilatore che il tipo *T* è un parametro. Ad esempio:

```
template<class T>
T C<T>::f(T& elementoDiTipoT)
{
  cout << "L'elemento vale ";
  cout << elementoDiTipoT;
}
```

Si noti come in ogni punto si ricordi che *T* è un parametro, e che la *classe C* è costruita usando il parametro *T*. Riportando il tutto al nostro esempio, relativo alla *classe Insieme*, avremo:

```
template<class E>
class Insieme
{
  public:
    //campi pubblici, ad esempio
    void Aggiungi(E);
  private:
    //campi privati, ad esempio
    E* primo; };
```

Le funzioni così dichiarate, saranno poi definite nel seguente modo:

```
template<class E>
void Insieme<E>::Aggiungi(E elemento)
{ //codice della funzione }
```

Nel momento in cui si farà uso della classe appena definita, si dovrà anche istanziare il tipo del parametro. Ad esempio:

```
Insieme<int> insiemeDiInteri;
Insieme<Scheda> schedario;
```

che sottintendono che i due insiemi sono, nell'ordine, un insieme di elementi di tipo *int* e del noto tipo *Scheda*. Una cosa dovrebbe essere evidente, a ben vedere i codici e il relativo formalismo: il tipo *T* può essere qualsiasi! Può essere una classe, oppure un tipo predefinito (es. *char*, *int*, etc.), oppure un tipo definito dall'utente (mediante una *typedef*).
Si noti che i template possono venire utilizzati anche in altri modi, non solo in abbinamento a classi. Ad esempio sono possibili dichiarazioni di tipi mediante template:

```
template<class C>
struct Elemento
{
  C info;
  Elemento* next;
};
```

oppure, si possono definire funzioni che fanno uso di template:

```
template<class T>
bool SonoUguali(T elem1,T elem2)
{ return elem1==elem2;}
```

Per un interessante esempio di uso dei template, vi rimandiamo al codice allegato. È proprio osservando il precedente frammento che si evidenzia una problematica nell'uso dei template. Se nella precedente funzione il tipo *T* fosse *int*, o un qualsiasi tipo predefinito, il confronto di uguaglianza tra i due elementi avrebbe certamente un senso (il C++ definisce automaticamente, per

i tipi predefiniti, tutti gli operatori). Ma cosa succederebbe se il tipo che fa da parametro fosse ad esempio un tipo definito dall'utente, o una classe creata da noi (ad es. Poligono)? Se l'operatore di confronto è stato ridefinito da colui che ha creato il tipo o la classe, allora non ci sono problemi. Se invece tale operatore non è stato definito (o ridefinito), allora davanti a noi si aprono tre strade:

1. si chiede a chi ha implementato la classe di aggiungere l'operatore (il che innesca una procedura molto laboriosa di release di aggiornamenti);
2. si estende la classe mediante ereditarietà, con tutto ciò che questo comporta;
3. si spera che l'operatore abbia un significato pur non essendo ridefinito.

Ovviamente, la terza strada è da sconsigliare.
Si noti, infine, che le considerazioni appena svolte valgono per ogni funzione definita mediante template (quindi, anche le funzioni membro di una classe) e per ogni operatore, non solo per quello di confronto di uguaglianza.

UN ESEMPIO

Vediamo ora di analizzare un esempio didattico, per capire meglio il funzionamento pratico dei template. Scriveremo una classe Pila, in grado di accettare come elementi dei generici elementi di tipo *T*, che saranno decisi solo in un secondo momento rispetto alla scrittura del template. In letteratura informatica, una pila è una particolare struttura dati, che può essere vista come un contenitore, con delle particolari operazioni per l'inserimento e l'estrazione di elementi. In particolare, si ha che l'ultimo elemento inserito nella pila (con l'operazione *Push()*) sarà sempre il primo ad essere estratto (con l'operazione *Pop()*). Questo fa un po' pensare a una pila di piatti da lavare: l'ultimo piatto sporco messo sopra tutti gli altri, sarà il primo ad essere preso per essere lavato (a meno di possedere una lavastoviglie). Supponendo di avere una pila di interi, la sequenza di operazioni:

```
Push(5) Push(2) Push(0) Pop() Pop()
```

Produrrà l'estrazione dei numeri:

```
0 2
```

in questo ordine, mentre il 5, che è stato il primo ad essere inserito, rimarrà nella pila. Ovviamente la pila è una struttura che si presta molto bene ad essere implementata tramite template, in quanto possiamo benissimo pensare ad un pila di interi, di char, di oggetti Scheda ecc. La definizione della struttura della classe si presenta così:

```
template <class T>
class Pila {
public:
        Pila(int);
        ~Pila();
        void Push(T);
        T Pop();
        int QuantiElementi();
        int Dimensione();
private:
        int dimensione;
        int num_elementi;
        T* vettore_elementi;
};
```

Come si può vedere, il tipo *T* è stato usato esattamente come se fosse un tipo predefinito o noto. Il costruttore di Pila prevede che si indichi la dimensione della pila (cioè il numero massimo di oggetti che potrà contenere); le operazioni *Push()* e *Pop()* fanno esattamente ciò che si è descritto in precedenza; la funzione *QuantiElementi()* restituisce il numero di elementi presenti nella pila, mentre *Dimensione()* restituisce il numero massimo di elementi che *Pila* può contenere. Le funzioni sono implementate nel seguente modo:

```
template<class T>
Pila<T>::Pila(int dim) {
dimensione =        dim;
num_elementi = 0;
vettore_elementi = new T[dimensione];
}
template<class T>
Pila<T>::~Pila() {
        delete[] vettore_elementi;}
template<class T>
void Pila<T>::Push(T elem) {
        if (num_elementi < dimensione) {
        vettore_elementi[num_elementi] = elem;
        num_elementi++;}
}
template<class T>
T Pila<T>::Pop() {
        if (num_elementi > 0) {
        num_elementi--;
return vettore_elementi[num_elementi];}
else
        return T();}
template<class T>
int Pila<T>::QuantiElementi() {
        return num_elementi;
}
template<class T>
int Pila<T>::Dimensione() {
        return dimensione;
}
```

Ovviamente, come già accennato in precedenza, si presuppone (per la funzione *Push()*) che sia definito il

La pila

In letteratura informatica, una pila è una particolare struttura dati, che può essere vista come un contenitore, con delle particolari operazioni per l'inserimento e l'estrazione di elementi. In particolare, si ha che l'ultimo elemento inserito nella pila (con l'operazione *Push()*) sarà sempre il primo ad essere estratto (con l'operazione *Pop()*). Questo fa un po' pensare a una pila di piatti da lavare: l'ultimo piatto sporco messo sopra tutti gli altri, sarà il primo ad essere preso per essere lavato.

costruttore di copia del tipo *T* che viene istanziato col template, in quanto questo è passato per valore e non per riferimento. È da notare, inoltre, che è stato implementato un insieme minimale di caratteristiche della classe Pila, altre funzioni potrebbero essere per esempio le seguenti:

```
bool Pila<T>::PilaVuota() {...};
bool Pila<T>::PilaPiena() {...};
void Pila<T>::SvuotaPila() {...};
//per inserire e estrarre pile da pile
Pila<T> Pila<T>::Pop(int dim) {...}; //estrae
                    //una pila di dimensione "dim"
void Pila<T>::Push(Pila<T> p) {...}; //inserisce
//una pila di oggetti T
```

oppure le classiche funzioni standard come il costruttore di copia e l'overload dell'operatore di assegnazione. Queste funzioni aggiuntive non presentano alcuna difficoltà concettuale, anzi si invita il lettore a implementarle, partendo dalla base di codice presente sul CD-Rom allegato.

UTILIZZO DELLA PILA

Nonostante la sua apparente semplicità, la pila risulta molto utilizzata in informatica e, a volte, si presta molto bene all'implementazione di algoritmi che, altrimenti, richiederebbero uno sforzo decisamente maggiore per essere realizzati. Riportiamo di seguito un piccolo programma che ha il compito di "girare" la stringa di caratteri che viene inserita, in modo da farla leggere al contrario. Con questo esempio mostriamo anche la corretta sintassi dell'utilizzo di una classe template all'interno del codice:

```
#include <iostream.h>
#include <strstrea.h>
#include "Pila.cpp"
#define DIMENSIONE 100
void main () {
  //istanzio una pila di char
  Pila<char> pila(DIMENSIONE);
  cout << "Inserisci una frase da invertire:\n";
  cout << "(Max " << DIMENSIONE << " caratteri)\n";
  //leggo una riga di testo da tastiera
  char* linea = new char(DIMENSIONE);
  cin.getline(linea,DIMENSIONE);
  istrstream is(linea);
  //inserisco ogni char nella pila
  while (!is.eof()) {
    char* c = new char;
    is.read(c,1);
    pila.Push(*c);
  }
  //elimino l'ultimo carattere spurio
  pila.Pop();
  //stampo la stringa invertita
  cout << "\nLa frase invertita e':\n";
  while (pila.QuantiElementi()>0)
  cout << pila.Pop();
  cout << "\n\n";
}
```

Come si vede, abbiamo istanziato una Pila di char in modo del tutto analogo a quanto illustrato in precedenza. Inserendo una stringa di caratteri da consolle e premendo invio, ogni carattere digitato viene inserito nella pila, tramite la funzione *Push()*, a partire dal primo. Successivamente viene chiamata la funzione *Pop()* in modo da "scaricare" tutta la pila. Ovviamente "usciranno" dalla pila i caratteri a partire dall'ultimo e si avrà quindi l'effetto desiderato di invertire la frase inserita. Un esempio di esecuzione del programma è il seguente:

Inserisci una frase da invertire:
(Max 100 caratteri)
Il re decide di cederli

La frase invertita e':

ilredec id edicved er lI

Una cosa da notare (e che magari può essere stata considerata una svista) è il fatto che, quando si usano i template, nonostante continui ad essere opportuna la divisione tra file header (*.h*) e file sorgente (*.cpp*), è necessario includere il file *.cpp* e non quello *.h*. Questo è dovuto al fatto che il compilatore, per generare il codice, deve conoscere quale è il tipo utilizzato dalla classe template. In pratica, compilando una classe template senza istanziare neanche una variabile di quel tipo, si genera un file di codice vuoto. Questo porta al fatto che non è possibile, volendo sviluppare una libreria software, compilare il codice e distribuire solo i file header e i file compilati, ma è necessario distribuire anche i sorgenti. Proprio per la "stranezza" di includere un file *.cpp* alcuni programmatori usano mettere definizione e implementazione di una classe template in un solo file (*.h*), e includere quello. Questa è solo una sottigliezza estetica, quindi in generale è possibile fare come più si preferisce.

CONCLUSIONI

Gli argomenti visti in questa lezione possono davvero contribuire a migliorare il codice prodotto rendendolo modulare e indipendente. Cosa forse ancor più importante è che, se assimilati bene, contribuiscono a migliorare anche lo stile di programmazione e, in definitiva, il programmatore stesso. Per questi motivi continueremo nella prossima lezione ad affrontare questo argomento e a sviscerarlo per renderlo un po' meno ostico di quanto può sembrare.
Non mancate!

Marco Del Gobbo
Alfredo Marroncelli

Contatta gli autori!

Se hai suggerimenti, critiche, dubbi o perplessità sugli argomenti trattati e vuoi proporle agli autori puoi scrivere agli indirizzi:

Alfredo
alfredo.marroccelli@libero.it

e

Marco
marcodelgobbo@libero.it

Questo contribuirà sicuramente a migliorare il lavoro di stesura delle prossime puntate.

MATLAB®

I PRIMI ALGORITMI

MATLAB

In questo nuovo appuntamento passiamo dall'uso interattivo a quello automatico ed iniziamo a usare gli algoritmi di base che ci saranno utili per esplorare i nostri dati in una maniera più evoluta.

File sul Web
www.itportal.it\iop68 \matlab2.zip

Quando usiamo MATLAB interattivamente assumiamo sostanzialmente di dover eseguire i nostri calcoli una sola volta; stiamo esplorando, non sappiamo dove ci condurrà il procedimento che stiamo seguendo, non siamo certi della sua correttezza, non vogliamo spendere tempo per formalizzare l'algoritmo in una maniera rigorosa. In altre parole, stiamo sperimentando tecniche e procedimenti per riservarci di decidere in un secondo tempo quale sia la strada da battere. Uno dei temi importanti di questo secondo passo nell'ambiente MATLAB riguarda la gestione di procedure che ci offrono un maggiore grado di automazione. Ora supponiamo di essere stati in grado di trovare un modo soddisfacente per visualizzare i nostri dati e non vogliamo più perdere tempo a digitare tutti i comandi. Quindi, abbiamo necessità di automatizzare il lavoro fatto fino a questo punto e di tenere maggiormente sotto controllo quello che faremo da ora in poi. In MATLAB possiamo usufruire della possibilità di scrivere porzioni di software che vengono chiamati "*script*". Uno script é un insieme sequenziale di comandi atti a replicare una sequenza di azioni che vogliamo tenere memorizzata. Lo script viene lanciato dalla *Command Window* di MATLAB e il comportamento delle variabili e degli algoritmi é identico a quello sperimentato nell'uso interattivo. Per costruire lo script dobbiamo utilizzare un'applicazione che possiamo vedere in Fig. 1. L'editor di MATLAB possiede innumerevoli funzioni che consentono di mantenere sotto controllo la scrittura e l'esecuzione dei programmi. In questa fase della nostra conoscenza dell'ambiente ci accontenteremo di usare questa applicazione come un semplice editor di testo per la scrittura di codice. Uno script MATLAB costruito per mezzo dell'editor deve essere salvato in una precisa directory. MATLAB vede un certo numero di directory e le considera come luoghi in cui andare a cercare i propri programmi. Per chiamare questa lista é sufficiente digitare:

Fig. 1: **L'editor di MATLAB viene invocato per mezzo del comando "edit".**

```
>> path
                MATLABPATH

        D:\MATLAB6p5\toolbox\matlab\general
        D:\MATLAB6p5\toolbox\matlab\ops
        D:\MATLAB6p5\toolbox\matlab\lang
        D:\MATLAB6p5\toolbox\matlab\elmat
        D:\MATLAB6p5\toolbox\matlab\elfun
        D:\MATLAB6p5\toolbox\matlab\specfun
        D:\MATLAB6p5\toolbox\matlab\matfun
        D:\MATLAB6p5\toolbox\matlab\datafun
        D:\MATLAB6p5\toolbox\matlab\audio
        D:\MATLAB6p5\toolbox\matlab\polyfun
        D:\MATLAB6p5\toolbox\matlab\funfun
        D:\MATLAB6p5\toolbox\matlab\sparfun
        D:\MATLAB6p5\toolbox\matlab\graph2d
        D:\MATLAB6p5\toolbox\matlab\graph3d
        D:\MATLAB6p5\toolbox\matlab\specgraph
        D:\MATLAB6p5\toolbox\matlab\graphics
        D:\MATLAB6p5\toolbox\matlab\uitools
        D:\MATLAB6p5\toolbox\matlab\strfun
        D:\MATLAB6p5\toolbox\matlab\iofun
        D:\MATLAB6p5\toolbox\matlab\timefun
        D:\MATLAB6p5\toolbox\matlab\datatypes
        D:\MATLAB6p5\toolbox\matlab\verctrl
        D:\MATLAB6p5\toolbox\matlab\winfun
        D:\MATLAB6p5\toolbox\matlab\winfun\comcli
```

MATLAB

```
D:\MATLAB6p5\toolbox\matlab\demos
D:\MATLAB6p5\toolbox\local
.
.
.
```

La lista di directory é modificabile ed é la funzione *"path"* stessa che se ne occupa. Per conoscere meglio quali siano le potenzialità di un comando usiamo il comando *"help"*; nel nostro caso otteniamo:

[1]

```
>> help path

 PATH Get/set search path.
    PATH, by itself, prettyprints MATLAB's current search
       path. The initial search path list is set by PATHDEF,
                                                and is perhaps
    individualized by STARTUP.
    P = PATH returns a string containing the path in P.
    PATH(P) changes the path to P.  PATH(PATH) refreshes
    MATLAB's view of the directories on the path, ensuring
   that any changes to non-toolbox directories are visible.
    PATH(P1,P2) changes the path to the concatenation of
      the two path strings P1 and P2.  Thus PATH(PATH,P)
          appends a new directory to the current path and
              PATH(P,PATH) prepends a new path.  If P1 or
           P2 are already on the path, they are not added.

    For example, the following statements add another
              directory to MATLAB's search path on various
                                         operating systems:
      Unix:     path(path,'/home/myfriend/goodstuff')
      Windows:   path(path,'c:\tools\goodstuff')

    See also WHAT, CD, DIR, ADDPATH, RMPATH,
                                          GENPATH, REHASH.
```

[1] Help

Il comando *help* consente di raggiungere la descrizione di un comando conoscendo esattamente il nome della funzione.

Così ora sappiamo che é anche possibile modificare a piacere la lista delle directory considerate da MATLAB come luoghi in cui andare a cercare i programmi. Esiste un modo interattivo di fare questa azione lanciando un'applicazione chiamata *"Set Path"* raggiungibile dal menu *"File/Set Path"* (Fig. 2).

[2] Lookfor

Il comando *lookfor* va alla ricerca di una data stringa contenuta nella prima riga dell'help di una funzione.

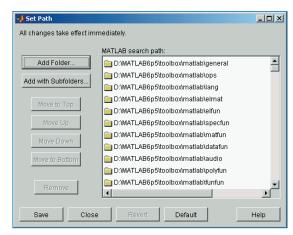



Fig. 2: Gestione del path di MATLAB.

Esiste ancora una directory, quella chiamata *"corrente"*; questa é la directory su cui é puntato MATLAB e puo' essere differente da quelle che compaiono nel path. Supponiamo per un momento di non sapere quale sia il comando che ce la mostra. Se siamo in questa situazione non siamo in grado nemmeno di usare il comando *"help"* poiché esso richiede di digitare il nome esatto del comando. Esiste allora una funzione che va alla ricerca di tutte le occorrenze di una stringa all'interno della prima riga dell'help di una funzione. Dall'help della funzione *"path"* notiamo che il formato della prima riga é: *nome della funzione* (in maiuscolo) e *breve descrizione della stessa*. Se ora immaginiamo che in questa breve descrizione vi debba essere qualche cosa di interessante possiamo eseguire una ricerca. MATLAB mette a disposizione la funzione *"lookfor"* per questo scopo. E' ora molto probabile che la parola *"directory"* sia contenuta nella descrizione della funzione che cerchiamo.

Quindi:

[2]

```
>> lookfor directory
ADDPATH Add directory to search path.
CD     Change current working directory.
COPYFILE   Copy file or directory.
DIR List directory.
DOCROOT A utility to get or set the root directory of
                                            MATLAB Help
ISDIR  True if argument is a directory.
LS List directory.
MKDIR   Make new directory.
MOVEFILE   Move file or directory.
PWD Show (print) current working directory.
RMDIR   Remove directory.
RMPATH Remove directory from search path.
WHAT List MATLAB-specific files in directory.
UIGETDIR Standard open directory dialog box
FILESEP Directory separator for this platform.
MATLABROOT Root directory of MATLAB installation.
PREFDIR Preference directory name.
TEMPDIR Get temporary directory.
.
.
.
```

Scopriamo così che il comando *"pwd"* fa al caso nostro:

```
>> pwd
ans =
d:\MATLAB6p5\work
```

La directory corrente è leggermente privilegiata sulle altre, essendo la prima su cui MATLAB va alla ricerca di programmi. Possiamo spostarci su una directory differente per mezzo del comando *"cd"* (*change directory*), come nell'esempio seguente:

```
>> cd IoProgrammo
```

```
>> pwd
ans =
d:\MATLAB6p5\work\IoProgrammo
```

Questo é un meccanismo che ci consente un ulteriore grado di libertà. Supponiamo di essere stati in grado di spostare la directory corrente di MATLAB sino a quella che contiene un file che ha nome *"coni.m"* (che é possibile trovare sul CD allegato alla rivista). Se ora digitiamo sulla *Command Window* il comando *"edit coni.m"* (l'estensione *".m"* é riservata ai programmi scritti in linguaggio MATLAB), invochiamo l'applicazione Editor.
Arrivati a questo punto possiamo modificare i dati, i parametri ed anche la sequenza di operazioni applicate ai nostri dati. Siamo pronti per eseguire lo script ed esplorare il risultato delle nostre operazioni. Per arrivare allo stesso punto in cui ci siamo lasciati nel numero precedente non dobbiamo fare altro che digitare sulla *Command Window* il comando *"coni"*. Il risultato di questa azione é quello di produrre una finestra MATLAB che riporta un insieme di coni per ogni settore di attività economica (Fig.3).

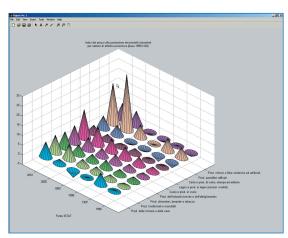



Fig. 3: Dal menu "Tools" selezionare "rotate3D" oppure premere il bottone presente sulla toolbar che riporta una freccia che ruota in senso antiorario, e muovere il grafico con il mouse per ottenere dei punti di vista alternativi.

Riferitevi al numero precedente per comprendere i dettagli implementativi di questo semplice script.
Dobbiamo ora necessariamente notare che i dati che abbiamo creato la volta scorsa in maniera interattiva e questa volta in maniera automatica devono risiedere da qualche parte. Questo luogo prende il nome di *"Workspace"*. Lo spazio di lavoro di MATLAB é un'area della RAM della nostra macchina gestita da MATLAB. É importante notare che non viene richiesta nessuna azione di allocazione diretta su questa area che non sia quella strettamente dipendente dall'esecuzione dei nostri comandi matematici. Noi manipoliamo oggetti matematici ma é MATLAB che si occupa di manipolare l'area di memoria per assicurare consistenza e solidità alle nostre operazioni.
Per vedere il workspace possiamo usare:

[3]

```
>> whos
```

Name	Size	Bytes	Class
categorie	9x1	1188	cell array
colonne	1x1	8	double array
colori	9x3	216	double array
dati	6x10	480	double array
h	1x1	8	double array
handle	1x1	8	double array
indiciPrezzi	6x9	432	double array
k	1x1	8	double array
n	1x1	8	double array
periodo	6x1	48	double array
r	1x1	8	double array
righe	1x1	8	double array
theta	1x16	128	double array
w	1x1	8	double array
x	2x16	256	double array
xx	2x16	256	double array
y	2x16	256	double array
yy	2x16	256	double array
z	2x16	256	double array

```
Grand total is 664 elements using 3836 bytes
```

Vedremo nel seguito che ci é utile conoscere il nome delle variabili e la loro dimensione (le prime due colonne) per essere in grado di scrivere dell'opportuno codice che ci aiuti a raggiungere il nostro successivo obbiettivo.

SISTEMI LINEARI

Questa volta il nostro scopo é la determinazione delle linee di tendenza dei prezzi per settore di attività economica, quindi abbiamo la necessità di introdurre brevemente un concetto matematico molto utile: l*a risoluzione di sistemi lineari*. Un sistema lineare é un insieme di equazioni lineari espresse in un certo numero di variabili; solitamente a noi interessa il valore che queste variabili devono assumere per soddisfare contemporaneamente tutte le equazioni. Prima di affrontare il nostro problema tentiamo di mostrare in maniera intuitiva i fondamenti di questi concetti per mezzo di un semplice esempio. Supponiamo che vi sia un viticultore astigiano che possiede 20 ettari di terreno che intende coltivare a Barbera. Egli acquista concime naturale da una fattoria vicina ed il suo costo per ettaro all'anno sara' uguale a 1000 Euro, che crescerà a 3000 Euro se utilizzerà del concime chimico. Un ettaro trattato a concime naturale produce 120 bottiglie all'anno vendute a 28 Euro l'una, mentre un ettaro concimato chimicamente produce il 50% in più e le bottiglie vengono vendute a 20 Euro l'una. L'investimento per i prossimi 5 anni é pari a 200,000 Euro. Il guadagno desiderato é pari al 30% in più rispetto all'investimento. Quanti ettari di terreno dovranno essere coltivati chimicamente e quanti in maniera naturale per rispettare i vincoli sopra citati? Prima di tutto scegliamo le variabili del problema in maniera appropriata. Chiamiamo *x1* il numero di ettari colti-

MATLAB

[3] Whos

Il comando *whos* elenca tutte le variabili presenti nel workspace menzionando nome, dimensione, byte occupati e classe dell'oggetto.

MATLAB

vati con concime naturale, $x2$ quelli concimati con concime chimico e $x3$ il numero di ettari non coltivati. Dal primo assunto sappiamo che la somma di tutti gli appezzamenti di terreno deve assommare a 20:

$$x_1 + x_2 + x_3 = 20$$

Spendendo 1000 Euro all'anno sul terreno concimato naturalmente, 3000 Euro all'anno su quello concimato chimicamente per cinque anni, dobbiamo ottenere la spesa totale pari a 200,000 Euro:

$$1000 \cdot 5x_1 + 3000 \cdot 5x_2 = 200000$$

Se ora desideriamo un ritorno pari a 260,000 Euro (+30%) dobbiamo scrivere:

$$120 \cdot 28 \cdot 5x_1 + 180 \cdot 20 \cdot 5x_2 = 260000$$

Se riflettiamo bene ora possediamo tre equazioni ma anche tre incognite. Possiamo tentare di risolvere il nostro sistema ricavando da una delle equazioni il valore di una delle variabili, sostituirlo nelle altre equazioni e proseguire in questa maniera sino a quando l'intero sistema non porta in luce i valori numerici di ogni singola variabile. Invece, esistono tecniche matematiche potenti che ci consentono di generalizzare il processo e risolvere sistemi anche più grandi con poco sforzo. Intanto possiamo esprimere il nostro sistema in forma matriciale per mezzo di:

$$\begin{pmatrix} 1 & 1 & 1 \\ 5000 & 15000 & 0 \\ 16800 & 18000 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 20 \\ 200000 \\ 260000 \end{pmatrix}$$

Se ricordiamo le regole di moltiplicazione matriciale mostrate nel numero precedente, scopriamo che questa espressione del nostro sistema ha perfettamente senso. Solitamente si assegna il nome "*A*" alla matrice dei coefficienti, "*x*" al vettore delle variabili da determinare e "*b*" al vettore colonna dei termini noti. Possiamo comparare la soluzione di un'equazione matriciale come la precedente ($Ax = b$) alla soluzione di un'equazione coinvolgente solo scalari come $7x = 10$:

$$7x = 10 \Rightarrow x = 7^{-1} * 10 = 10/7$$

Ricordiamo che 7 elevato alla –1 equivale a 1/7. Allo stesso modo possiamo esprimere la nostra equazione matriciale (il nostro sistema di equazioni) con:

$$Ax = b \Rightarrow x = A^{-1} * b = A\backslash b$$

Qui abbiamo usato l'operatore *backslash* (\) per eseguire quella che viene detta moltiplicazione sinistra per A^{-1} (l'inversa di A). Non abbiamo potuto usare il segno di divisione destra (/) come per gli scalari, poiché le matrici seguono regole un po' più complesse.

Ricordiamo che mentre la moltiplicazione tra scalari é commutativa ($a*b$ é identico a $b*a$), la moltiplicazione tra matrici non lo é ($A*x$ non é uguale a $x*A$ a meno di un caso particolare che abbiano discusso nel numero precedente). Abbiamo scoperto ora un operatore particolarmente utile in MATLAB. Se definiamo le variabili fondamentali del nostro problema con:

[4]

```
>> A = [1 1 1; 1000*5 3000*5 0; 120*28*5 180*20*5 0]
A =
          1        1        1
          5000     15000    0
          16800    18000    0

>> b = [20 200000 260000]'
b =
          20
          200000
          260000
```

Sappiamo che la soluzione (il valore delle variabili incognite) può essere calcolato per mezzo di:

[5]

```
>> x = A \ b
x =
1.8519
12.7160
5.4321
```

Possiamo concludere dicendo che il nostro viticultore dovrà coltivare a uve Barbera circa 1.8 ettari con concime naturale, circa 12.7 ettari con concime chimico e potrà lasciarne incolti circa 5.4. Un interessante tentativo é quello che possiamo fare provando a variare i prezzi del vino di migliore qualità e di quello di qualità inferiore per vedere come varia la dimensione di terreno assegnato all'uno o all'altro tipo di coltivazione.

LINEE DI TENDENZA

Il nostro prossimo compito é quello di calcolare le linee di tendenza per ogni settore di attività economica. Per fare questo dobbiamo calcolare, per ogni settore, la retta che meglio approssima i dati. Per ottenere questo risultato possiamo costruire una retta (di equazione: $y=ax+b$) passante per ognuno dei punti che abbiamo a disposizione e mettere tutte queste equazioni in un sistema. Con lo scopo di individuare la migliore accoppiata di parametri "*a*" e "*b*" (detti *coefficiente angolare* e intercetta sull'asse *y*) che individuano una retta approssimante l'andamento dei dati. Il sistema che ne risulta possiederà soltanto due incognite e molte equazioni e viene normalmente chiamato *sovradeterminato*. Vale a dire che possediamo più leggi di comportamento delle variabili di quante siano le variabili stesse. Esaminiamo alcune caratteristiche salienti dei sistemi sovradeterminati e sulla loro soluzione. Nel nostro caso specifico se tentiamo di scrivere tutte le equazioni delle rette che passano per

[4] Sintassi

Spazi o virgole separano le colonne. I punti e virgola separano le righe. L'apice associato ad un vettore o ad una matrice opera una trasposizione.

[5] Help

Nella Command Window di MATLAB digitare "help \" oppure "help mldivide"

tutti i punti del primo andamento, otteniamo:

$1996a + b = 0.7$
$1997a + b = 6$
$1998a + b = 0.1$
$1999a + b = -3.6$
$2000a + b = 2.4$
$2001a + b = 7.9$

Già sappiamo che possiamo esprimere il sistema in forma matriciale per mezzo di:

$$\begin{pmatrix} 1996 & 1 \\ 1997 & 1 \\ 1998 & 1 \\ 1999 & 1 \\ 2000 & 1 \\ 2001 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 0.7 \\ 6 \\ 0.1 \\ -3.6 \\ 2.4 \\ 7.9 \end{pmatrix}$$

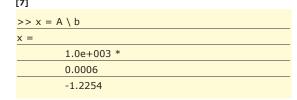
$$A \cdot x = b$$

Quando un sistema possiede più vincoli (equazioni) di quanti siano i gradi di libertà (variabili) solitamente non ha soluzione, a meno che alcune equazioni siano delle combinazioni lineari (riscritture) delle altre equazioni. Abbiamo un modo per misurare il numero di equazioni che non siano combinazioni lineari delle altre e questo é il *rango*. Un sistema di equazioni nella forma $Ax = b$ ha soluzioni se e solo se il rango della matrice A é identico a quello della matrice formata dall'accoppiamento della matrice A e del vettore b (in MATLAB scriveremo *[A, b]*). Il sistema che appare sopra viene rappresentato da una matrice di coefficienti e da un vettore colonna di termini noti:

```
>> A = [periodo ones(size(periodo,1),1)]
A =
        1996     1
        1997     1
        1998     1
        1999     1
        2000     1
        2001     1

>> b = indiciPrezzi(:,1)
b =
        0.7000
        6.0000
        0.1000
        -3.6000
        2.4000
        7.9000
[6]
>> rank(A) - rank([A,b])
ans =
        -1
```

Il $rank(A) = 2$, ma $rank([A, b]) = 3$ quindi il sistema non ha soluzioni esatte e nel nostro sistema di sei equazioni ne necessitano solo tre per descrivere pienamente la retta cercata; evidentemente vi sono tre equazioni che sono combinazioni lineari delle altre. Il sistema continua ad essere sovradeterminato poiché abbiamo due variabili ma tre equazioni tra loro indipendenti. Ricercare quali siano le equazioni linearmente dipendenti può essere un compito matematicamente gravoso. Comunque, MATLAB é in grado di risolvere questo sistema:

```
[7]
>> x = A \ b
x =
        1.0e+003 *
        0.0006
        -1.2254
```

In realtà, nel caso di sistemi sovradeterminati, l'operatore backslash \ risolve un problema di ottimizzazione:

$$\min \left\| Ax - b \right\|$$

Cioè, viene trovata una x che soddisfi la regola che la norma della differenza tra Ax e b sia la più piccola possibile. In termini più intuitivi possiamo dire che l'algoritmo trova dei parametri incogniti per fare in modo che la distanza tra la retta cercata ed i singoli punti sia la più piccola possibile. Abbiamo quindi scoperto un secondo comportamento dell'operatore backslash. In MATLAB esso viene utilizzato in moltissimi casi di risoluzione di sistemi lineari ed è in grado di adattarsi automaticamente ai dati che gli vengono sottoposti per trovare la soluzione più appropriata. Nel nostro caso particolare abbiamo utilizzato una retta per approssimare l'andamento dei nostri dati, ma nulla ci vieta di utilizzare un altro tipo di funzione (rispettando il vincolo di linearità della sua forma) che ci sembri più adatta a descrivere il fenomeno che stiamo analizzando. Ora abbiamo bisogno di calcolare in un numero appropriato di punti la nostra retta di cui conosciamo ora il coefficiente angolare e l'intercetta sull'asse y. Qui di proposito abbiamo utilizzato un numero di punti diverso da quelli da cui eravamo partiti per calcolare i coefficienti della retta. Questo viene fatto per mostrare che abbiamo il pieno controllo dei dati e dell'applicazione degli algoritmi ai casi reali che incontriamo.

```
[8]
>> x_retta = [periodo(1):0.5:periodo(end)]
x_retta =
  1.0e+003 *
  Columns 1 through 10
    1.9960    1.9965    1.9970    1.9975    1.9980
          1.9985    1.9990    1.9995    2.0000    2.0005
  Column 11
    2.0010
>> y_retta = x(1)*x_retta + x(2)
y_retta =
  Columns 1 through 10
    0.7143    1.0214    1.3286    1.6357    1.9429
```

[6] help rank
Nella Command Window di MATLAB digitare "help rank"

[7] fattore di scala
Il fattore di scala 1.0e+003 viene utilizzato per rendere più concisi e leggibili i risultati.

[8] :
L'operatore ":" ci consente di generare automaticamente una sequenza di numeri.

Bibliografia

• FONDAMENTI DI CALCOLO NUMERICO *Giovanni Monegato* (Edizioni CLUT) 1998

• METODI NUMERICI E STATISTICI PER LE SCIENZE APPLICATE *Valeriano Comincioli* (Editrice Ambrosiana) 1992

MATLAB

[9] Hold on

Di norma ogni comando di visualizzazione fa scomparire il precedente grafico per lasciare posto al nuovo; la funzione "hold on" lo impedisce, facendo in modo che ogni nuovo grafico si vada semplicemente ad aggiungere a quelli esistenti.

[10] Comando grafico

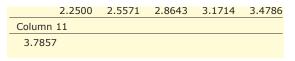
Un comando grafico ha solitamente la capacità di restituire un handle: il riferimento all'oggetto grafico appena creato. Puo' essere utilizzato per variare o interrogare le sue proprieta'.

Sul Web

Getting Started with MATLAB
http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/getstart.pdf

Using MATLAB
http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/using_ml.pdf

Using MATLAB Graphics
http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/graphg.pdf

```
    2.2500    2.5571    2.8643    3.1714    3.4786
  Column 11
    3.7857
```

Quindi visualizziamo il risultato:

[9]

```
>> plot(periodo, b, '+r')
>> hold on
>> plot(x_retta, y_retta)
```

ed otteniamo la Fig. 4.

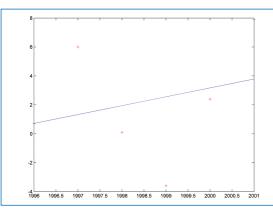
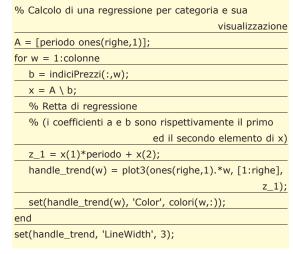



Fig. 4: I punti originali sono in rosso mentre la retta é quella che ha la minima distanza possibile da ogni punto.

Il nostro compito é ora quello di ripetere lo stesso procedimento per tutte le attività economiche e fare in modo che le linee di tendenza appaiano sul grafico tridimensionale dove i prezzi sono già rappresentati per mezzo dei coni. Sappiamo che possiamo utilizzare uno script e andiamo quindi a scrivere il codice seguente all'interno del nostro editor.

[10]

```
% Calcolo di una regressione per categoria e sua
                                        visualizzazione
A = [periodo ones(righe,1)];
for w = 1:colonne
    b = indiciPrezzi(:,w);
    x = A \ b;
    % Retta di regressione
    % (i coefficienti a e b sono rispettivamente il primo
                               ed il secondo elemento di x)
    z_1 = x(1)*periodo + x(2);
    handle_trend(w) = plot3(ones(righe,1).*w, [1:righe],
                                                    z_1);
    set(handle_trend(w), 'Color', colori(w,:));
end
set(handle_trend, 'LineWidth', 3);
```

Salviamo il codice su un file che chiameremo *"tendenza.m"* e dopo averlo eseguito otterremo il risultato riportato in Fig. 5. Nel codice contenuto nel programma *"tendenza.m"* possiamo ora districarci senza eccessivi problemi. Abbiamo definito la matrice *"A"* così come avevamo fatto nell'esempio interattivo precedente. Il ciclo viene usato per scandire le colonne della matrice *"indiciPrezzi"*. Infatti, all'interno del ciclo

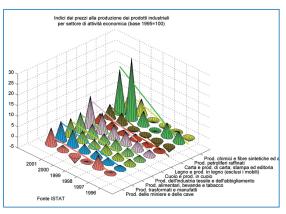



Fig. 5: Abilitate la Toolbar relativa alla gestione della Camera con il menu *"View/Camera Toolbar"* e tentate di modificare interattivamente la visualizzazione del grafico.

viene creato un nuovo vettore colonna *"b"* (i punti sull'asse x non cambiano, mentre i prezzi cambiano per ogni settore di attività economica che prendiamo in considerazione). Siamo ora in grado di calcolare per mezzo dell'algoritmo MATLAB "backslash" il valore dei parametri della retta. Una volta che i parametri sono noti possiamo calcolare la retta stessa nei punti contenuti nel vettore *"periodo"*, quindi usiamo questa retta per disegnare un segmento sul grafico tridimensionale creato in precedenza. Durante questa operazione chiediamo anche che ci venga restituito l'handle del segmento in modo che sia possibile usarlo per variare la sua proprietà di colore e ottenere così un segmento dello stesso colore del cono corrispondente. L'ultimo comando al di fuori del ciclo *"for"* ci serve per cambiare lo spessore delle linee di tendenza e renderle più spesse. Notiamo che abbiamo avuto l'accortezza di creare un vettore che contenga tutti gli handle di tutte le linee di tendenza.
In questo modo ora possiamo imporre a tutti gli handle contenuti nel vettore un cambiamento per mezzo di un solo comando (in maniera vettorizzata).

CONCLUSIONI

In questo numero abbiamo imparato a mescolare la modalità interattiva con quella procedurale di MATLAB. Questo ci consente di adattare il nostro comportamento al mutare delle nostre esigenze e ci consente di essere flessibili ed adattabili alla tipologia di problema da risolvere. Inoltre, siamo venuti in contatto con alcuni degli algoritmi di base ma di estrema potenza che sono a disposizione all'interno dell'ambiente MATLAB. Per maggiori informazioni sui prodotti della famiglia MATLAB potete consultare il sito di The MathWorks (*www.mathworks.it*). Nei prossimi numeri approfondiremo la conoscenza dell'ambiente da un punto di vista procedurale e questo ci consentirà di utilizzare in maniera comoda tutta la potenza di calcolo numerico che abbiamo a disposizione in MATLAB.

Fabrizio Sara (fsara@mathworks.it)

Ambienti renderizzati parte seconda

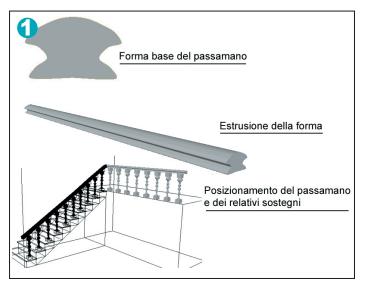
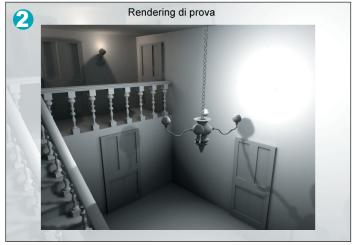
▶ Lightwave

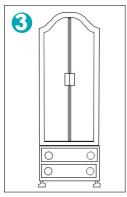
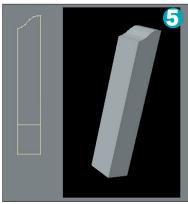
Roberto Lombardo

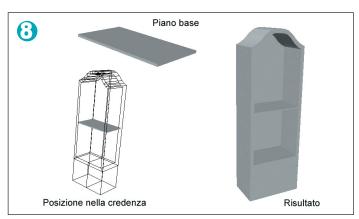
Proseguiamo la realizzazione della nostra stanza: completiamo il modello base e il mobilio principale.

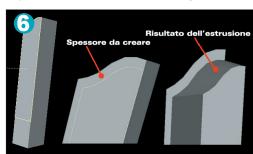
Nello scorso numero abbiamo creato la forma base della nostra stanza: le pareti, il pavimento, il soffitto, la scalinata principale, le porte e il lampadario centrale. In questa seconda parte del tutorial completeremo la parte superiore del passamano e creeremo alcuni mobili che andranno ad arredare la nostra stanza. Tutti questi elementi saranno successivamente texturizzati per la resa finale. Essendo una stanza pre renderizzata non abbiamo limiti di poligoni o di dimensioni e numero di texture, poiché il risultato finale che ci serve è quello in fase di render. Quando realizzate una stanza di questo tipo il mio personale consiglio è di procurarsi il maggior numero di riferimenti possibili, in modo da avere ben chiaro in mente il risultato finale che vogliamo raggiungere, oggi la Rete ci offre tutto ciò di cui abbiamo bisogno, siano esse schermate provenienti da videogame che da rendering architettonici o foto d'interni. In questo caso la fonte d'ispirazione principale è Resident Evil, famoso videogame 3D con ambienti pre renderizzati che ha fatto scuola in questa categoria di videogiochi. Non ci resta che aprire il nostro programma 3D e riprendere il lavoro da dove è stato interrotto.

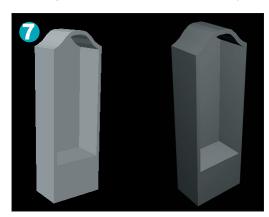
▼1-2 Completiamo il passamano della scala al piano superiore

Dobbiamo ultimare la nostra scala completando il passamano del piano superiore, il procedimento è semplice, prendiamo la forma base dalla quale siamo partiti per l'estrusione del passamano (Fig.1) quindi estrudiamola orizzontalmente per la lunghezza stabilita sino a toccare la parete dal lato opposto. Ora sistemiamolo lungo il piano superiore: potremmo già posizionare le basi del passamano come in figura, ma ricordiamoci di rimuoverle in seguito, poiché ne texturizzeremo soltanto una e la cloneremo risparmiando cosi molto tempo. Adesso con il passamano al completo, la nostra stanza "base" è pronta, ora dobbiamo dedicarci al mobilio che andrà ad arredare e completare il nostro modello "grezzo". Per questo inseriremo una credenza, una poltroncina, una mensola, qualche quadro. In Fig. 2 possiamo vedere un rendering temporaneo dell'oggetto per visionare la posizione del passamano superiore appena creato.

3-4-5 La credenza

Creiamo adesso il modello della credenza. In Fig. 3 abbiamo un semplice schizzo della credenza che dobbiamo modellare. Importiamo questo schizzo nel nostro programma 3d come Background in una viewport e modelliamoci sopra. Se fosse necessario invertiamo i colori dello schizzo per una migliore visibilità. Partiamo dalla forma vera e propria del mobile, in Fig. 4 abbiamo il nostro schizzo in background e la forma base piana che vi creeremo sopra. Basterà modellare solo una metà del mobile per poi eseguire un mirror dell'oggetto. Una volta creata la forma base eseguiamo una suddivisione all'altezza dei cassettoni come evidenziato in Fig. 5.

8-9 Mensole e vetri

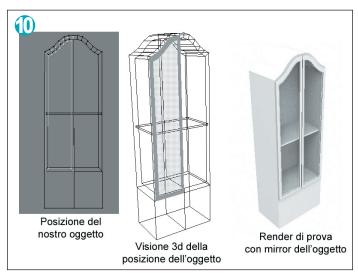
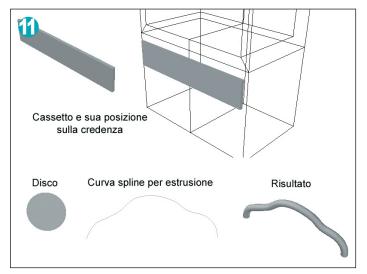
Ora creiamo delle semplice mensole da inserire all'interno della nostra credenza, un semplice box strecciato fa al caso nostro. Inseriamole soltanto, successivamente. Una volta texturizzata, la cloneremo per creare le altre, il risultato dell'operazione è in Fig. 8. Realizziamo adesso i due sportelli con vetro per chiudere la parte superiore della nostra credenza. Anche in questo caso ne creeremo uno e lo specchieremo con un mirror per risparmiare tempo, sempre dopo averlo texturizzato. Selezioniamo i vertici esterni della parte superiore della nostra credenza come mostrato in Fig. 9, copiamoli in un altro layer (Lightwave) e creiamo dagli stessi il nuovo poligono, diversamente possiamo selezionare i stessi vertici e creare il poligono prima, rimovendolo e spostandolo in un secondo momento per effettuare le altre operazioni. Una volta creato il poligono base selezioniamo nuovamente i vertici ed estrudiamoli creando così il bordo del nostro oggetto. Così facendo il poligono interno grande diverrà automaticamente il vetro della nostra credenza (Fig.9).

6 Estrusione

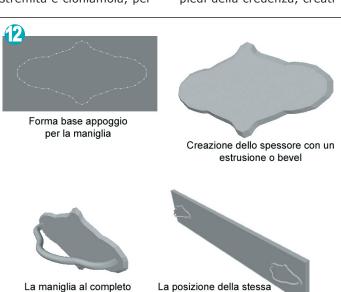
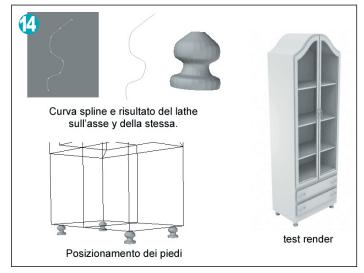
Creata questa suddivisione eseguiamo un'estrusione della forma base ottenendo cosi lo spessore della nostra credenza. Ora dobbiamo creare l'interno della parte superiore della nostra credenza compreso un lieve spessore del legno, questa operazione risulta differente da programma a programma, l'operazione da compiere è un'estrusione interna del poligono frontale superiore della nostra credenza, evidenziato in Fig. 6, estrusione che deve creare al tempo stesso uno spessore minimo, date un'occhiata alla Fig. 6.

7 Mirror

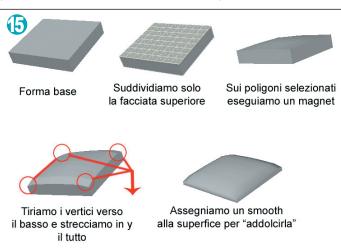
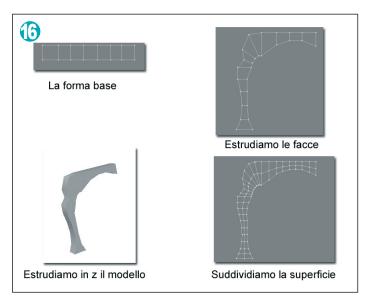
Eliminiamo eventuali poligoni in eccesso generati dall'estrusione e i poligoni interni presenti sul lato da specchiare. Una volta eseguito questo passaggio possiamo eseguire il mirror dell'oggetto ottenendo cosi il risultato mostrato in Fig. 7.

Ora basterà selezionare i poligono della fascia esterna appena creata ed estruderli in Z per creare lo spessore del legno, il risultato finale di tale operazione e tutti i passaggi sono rappresentati in Fig. 9.

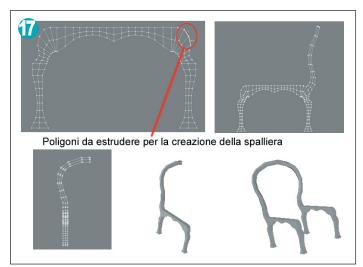
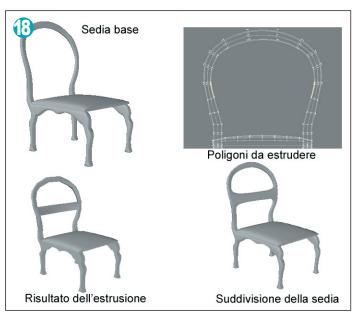
10-11 I cassetti

Adattiamo l'oggetto appena creato alla credenza come in Fig. 10.
Se vogliamo, possiamo eseguire un mirror di prova dell'oggetto per verificare il risultato. Adesso per completare il nostro oggetto dobbiamo creare 2 cassetti "fittizi" da posizionare nella parte bassa della stessa.

Per la forma del cassetto utilizzeremo il solito cubo strecciato con un lieve spessore come mostrato in Fig. 11.
Per creare le maniglie del cassetto estruderemo una forma base lungo una spline, in Fig. 11 possiamo vedere un disco estruso lungo una spline creando così una delle maniglie.

12-13-14 Ultimi ritocchi alla credenza

Adesso creiamo la base, una forma poligonale come quella rappresentata in Fig. 12 può andare bene, estrudiamola creando così un lieve spessore e posizioniamo la maniglia su di essa esattamente come in Fig. 12. Posizioniamo la maniglia sul cassetto ad un'estremità e cloniamola, per eseguire un render di prova possiamo clonare il cassetto al completo, un test di render è visibile in Fig. 13. Mancano ancora due piccoli elementi alla nostra credenza, le maniglie per gli sportelli con vetro, maniglie che cloneremo dai cassetti escludendo la base, infine i piedi della credenza, creati tramite un lathe (rivoluzione) di una curva spline attorno all'asse y.
In Fig. 14 abbiamo i 2 elementi e un render completo dell'oggetto. Adesso la credenza è completa, passiamo a realizzare la poltroncina per poi concludere gli elementi di base con una mensola e alcuni quadri qui è la.

▼15-16 La poltroncina

Creiamo un box come quello in Fig. 15, eseguiamo una suddivisione della facciata superiore come mostrato in figura, quest'ulteriore suddivisione diverrà la parte morbida del cuscino della nostra poltrona.Una volta suddivisa la facciata dobbiamo sollevare la parte centrale dei poligoni creando così il cuscino vero e proprio. Con Lightwave questo si ottiene effettuando un "magnet" sulla zona interessata. Fatto questo accentuiamo la forma del cuscino tirando verso il basso i 4 vertici all'estremità di ogni angolo, li trovate evidenziati in figura. Adesso dobbiamo creare la base in legno con i piedi, partiamo da una forma base piana come quella in Fig. 16, estrudiamo di volta in volta i poligoni sino a creare la forma evidenziata in figura.

▼17-18 Addolcire con Metanurbs

Eseguiamo una suddivisione delle superfici per addolcire le forme, estrudiamo in z la forma appena creata, creando così lo spessore dell'oggetto, quindi facciamo un mirror dell'oggetto ottenendo così la forma voluta, visibile in Fig. 17. A questo punto dobbiamo estrudere da uno dei due lati la spalliera della nostra poltrona, basterà selezionare il poligono interessato ed estruderlo di volta in volta, ritoccando i vertici sino ad ottenere un risultato simile a quello mostrato in Fig. 17. Una volta creata la spalliera possiamo specchiare l'oggetto, la sedia cosi sarà quasi completa. Adattiamo adesso il cuscino creato alla nostra sedia "strecciandolo" di volta in volta sino ad ottenere la dimensione esatta come rappresentato in Fig. 18. Se vogliamo, possiamo eseguire un'ulteriore smooth della sedia per addolcire maggiormente le forme, per gli utenti di Lightwave basterà attivare le **Metanurbs** solo sulla sedia. Per completare la nostra poltrona dobbiamo solo creare un'ulteriore barra di legno al centro della spalliera estrudendo i poligoni interni centrali della stessa come evidenziato in Fig. 18. Adesso anche la nostra semplice poltroncina è completa, possiamo passare alla creazione di una piccola mensola da applicare ad una delle pareti.

▼19 La mensola

Quest'oggetto è molto semplice da realizzare, dobbiamo creare un box di base "strecciato" in x come quello in fig. 19, quindi estrudiamo di poco la facciata superiore e strecciamola lievemente in modo da renderla più piccola ottenendo il risultato mostrato in Fig. 19. Per creare i bracci che reggono l'oggetto utilizziamo la solita tecnica dell'estrusione lungo una spline, creiamo un disco ed estrudiamolo lunga una come mostrato in Fig. 19, quindi posizioniamolo ad una estremità della nostra mensola e specchiamolo ottenendo così anche il suo corrispondente dall'altra parte. Con questo passaggio abbiamo completato anche questo oggetto, non ci resta che creare qualche quadro da inserire in scena, la creazione della cornice consiste un semplice estrusone e smooth di una cornice di poligoni, la parte relativa alla tela sarà un semplice piano inserito nella cornice.

▼20-21 Il quadro

Partiamo col realizzare una cornice poligonale come quella in Fig. 20. Per comodità lavoreremo solo su una sezione della cornice che specchieremo in seguito. Selezioniamo la fila di vertici interni e tiriamoli verso il basso creando così una rientranza nella struttura. Adesso selezioniamo i vertici esterni della nostra cornice ed estrudiamoli verso l'esterno come visibile in figura.
Cosi facendo creiamo una nuova fila di poligoni da estrudere creando lo spessore laterale della nostra cornice, per meglio comprendere le operazioni da seguire date un'occhiata alla Fig. 20. Anche il quadro è quasi completo, il risultato della creazione dello spessore è visibile in Fig. 21, adesso possiamo specchiare il nostro modello creando così la mesh completa, se necessario mergiamo i punti dopo il mirror per evitare doppi vertici. Ora possiamo creare un semplice piano da applicare come tela del nostro quadro ottenendo il risultato in Fig. 21, se fosse necessario possiamo addolcire le forme del quadro con uno smooth sullo stesso aumentandone di conseguenza i poligoni.

▼22 Considerazioni

Abbiamo in render di prova del nostro ambiente in Fig. 22, se tutto è andato per il verso giusto dovreste avere un modello simile a questo. Naturalmente non c'è limite al numero di elementi che possiamo inserire in scena per aumentare il dettaglio: vasi, tappeti, oggetti dettagliati, e quant'altro ci venga in mente aumenterà il realismo e la definizione della nostra stanza.
Gli elementi presentati in questo tutorial sono solo pochi e semplici oggetti, realizzati al solo scopo di dare una traccia di partenza per la realizzazione di un modello di stanza come questo, una volta appresi i meccanismi base di modellazione di oggetti per un ambiente come questo l'unico limite sarà il tempo a vostra disposizione e la ram del vostro computer. Gli oggetti presentati in questo articolo sono molto semplici da realizzare ogni elemento può essere creato in maniera identica o quasi con qualsiasi programma 3d oggi in commercio. Questo è un motivo per cui abbiamo scelto oggetti semplici e immediati da modellare con tecniche basilari come estrusioni e forme poligonali di base, senza perderci in inutili approfondimenti sicuramente inopportuni per questo articolo. Lo scopo è di procedere per gradi, una volta realizzata interamente la stanza, texturizzata e renderizzata ottenendo un risultato sicuramente di buon livello possiamo passare ad oggetti e stanze più complesse avendo appreso ormai le basi di modellazione non organica.

Introduzione
AL PATTERN RECOGNITION

Con il termine di pattern recognition (PR) si identifica un insieme di approcci informatici che si occupano di attribuire un significato a insiemi di informazioni (dette pattern o segnali di ingresso).

I primi studi di PR risalgono agli anni '60; benché non sia ancora stata sviluppata una teoria unificata come per altre discipline delle scienze dell'informazione, sono stati raggiunti risultati scientifici molto interessanti e svariate applicazioni pratiche sono state sviluppate. Basti pensare ai sistemi automatici per il riconoscimento di caratteri, ai sistemi di navigazione per robot autonomi in ambienti sconosciuti, all'analisi automatica di immagini in campo astronomico, medico… gran parte dei fenomeni dell'esistenza umana si manifestano sotto forma di pattern: i simboli della scrittura, gli elementi costitutivi del parlato, i volti delle persone,… Gli stessi esseri umani valutano le situazioni in termini di pattern e agiscono in base a queste valutazioni: possiamo affermare che il PR è una componente del comportamento umano. Obiettivo ultimo di ogni ricercatore che lavori nell'ambito del PR è costruire macchine che possiedano le nostre stesse capacità di riconoscimento di pattern.

INTRODUZIONE AL PR

La traduzione letterale di Pattern Recognition è "Riconoscimento di Forme", ma la traduzione di pattern con forme è alquanto riduttiva. In maniera "poetica" possiamo utilizzare la definizione del Watanabe, il quale definisce un pattern come l'opposto del caos e come un'entità vagamente definita cui può essere dato un nome. Ad esempio un pattern può essere un volto, un carattere scritto a mano, un'impronta digitale, un segnale sonoro, l'andamento di un titolo di borsa… In definitiva il termine pattern indica l'oggetto utilizzato per rappresentare il fenomeno chi si sta osservando. Nel contesto del Pattern Recognition la classificazione è intesa come assegnamento del pattern a una classe. Per classe intendiamo un insieme di entità aventi proprietà comuni (ad esempio i diversi modi in cui le persone scrivono il carattere "A").
Diamo qualche definizione rigorosa:

"La classificazione si dice supervisionata nel caso in cui le classi sono note a priori e i pattern di esempio sono etichettati (cioè conosco la loro classe di appartenenza); non-supervisionata nel caso in cui le classi sono sconosciute e devono essere derivate dai dati stessi."
"Se un pattern viene associato a una sola classe si parla di classificazione esclusiva; se invece il pattern può appartenere, con un certo grado di probabilità, a più classi si parla di classificazione continua."

La grande importanza del PR è dovuta al fatto che

IL SISTEMA DI PR

Vediamo ora come può essere schematizzato un generico sistema di PR riportato in Fig. 1. Un dispositivo di acquisizione, (ad esempio una telecamera), produce i pattern, questi vengono sottoposti ad una fase di preprocessing che è necessaria al fine di normalizzare i dati di appartenenza per renderli adatti ad una successiva analisi. In genere sono utilizzate tecniche di enhancement per ridurre la presenza di rumore e più in generale trasformazioni che rendono le successive fasi più semplici ed efficaci. Il passo successivo consiste nell'estrazioni delle feature. Informalmente una feature può essere definita come un qualsiasi tipo di misura estraibile. Ciò che si cerca di ottenere con questa operazione è un maggior potere discriminante unito ad

Fig. 1: Schema di un generico sistema di pattern recognition

Pattern Recognition

File sul CD
\soft\codice\prtools

Obiettivi
Lo scopo di un sistema di PR è quello di fornire una descrizione dei dati forniti, tale descrizione consiste nella classificazione dei pattern in una serie di categorie conosciute.

Pattern
Recognition

Problema	Applicazione	Input	Output
Analisi di documenti	OCR	Immagini di documenti	Caratteri, parole
Automazione Industriale	Ispezione di circuiti stampati	Immagine del circuito	Difettoso / Non Difettoso
Bioinformatica	Analisi delle sequenze	Sequenze DNA / Proteine	Tipo di gene / Proteine
Classificazione di documenti	Ricerca su internet	Documento	Categoria
Database Immagini	Ricerca di immagini	Collezione di immagini	Specifici soggetti o temi
Economia	Predizione mercato azionario	Sequenze storiche	Acquista / vendi
Medicina	Analisi immagini radiografiche	Immagini	Sano / malato
Militare	Abbattimento automatico missili	Immagini	Direzione di calibrazione
Riconoscimento del parlato	Risponditori telefonici automatici	Segnale audio	Parole dette
Sistemi Biometrici	Riconoscimento di persone	Volto, Iride, Impronta digitale	Utente autorizzato
Sorveglianza	Sistema anti - intrusione	Immagini "live" del locale	Normale / Allarme
Telerilevamento	Stimare densità di colture	Immagini multispettrali	Tipi di coltivazioni
Visione robotica	Guida automatica di un veicolo	Immagini "live"	Direzione sterzo

Tab. 1: **Applicazione di Pattern Recognition.**

una riduzione dei dati da fornire in ingresso al classificatore. Le feature possono essere numeriche o simboliche; in generale si tratta comunque di entità di alto livello, in quanto l'idea è quella di catturare l'informazione utile in pochi dati. È però necessario considerare che la riduzione dei dati può comportare un aumento della probabilità di perdere informazioni utili, si dice che un insieme di feature è *ottimale*, quando non comporta un aumento della probabilità di errore nella classificazione. L'ultima fase consiste nella classificazione, ossia nel determinare la classe di appartenenza del pattern in ingresso a partire dalle feature estratte.

PRINCIPALI APPROCCI DEL PR

Il classificatore è il cuore di ogni sistema di PR. Storicamente i quattro principali approcci alla classificazione sono:

Template matching - Per capire questo metodo facciamo l'esempio di dover riconoscere un volto. Per ottenere l'identità di un'immagine la confronteremo con quelle che abbiamo. Se tale confronto lo facciamo semplicemente utilizzando la distanza diretta fra le due immagini le cose non funzionano (Fig. 2).
Il mancato funzionamento è dovuto al fatto che le varie immagini sono traslate, ruotate, scalate, esistono cambiamenti di illuminazione… L'idea è costruire uno o più pattern modello (template) (Fig. 3) e cercarlo all'interno dell'immagine (Fig. 4) misurando il grado di matching nelle diverse posizioni.

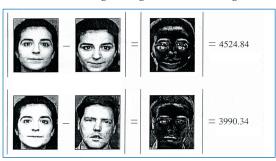



Fig. 2: **Misura delle distanze fra due immagini.**

Fig. 3: **Le immagini di partenza.**

Fig. 4: **La ricerca del Matching.**

APPROCCIO STATISTICO

Assumendo che da ogni pattern siano estratte n feature di tipo numerico, è possibile rappresentare ogni pattern come un vettore di features. Come dice il nome tale approccio si basa su metodi statistici, in particolare basati sulla teoria della decisione di Bayes. Il più classico esempio di approccio statistico è costituito dalla regola Nearest Neighbor (sarà trattato in maniera più approfondita nel prossimo articolo). Tale metodo si fonda sulla ragionevole assunzione che pattern fra loro vicini secondo una qualche metrica , appartengano alla stessa classe. Dati un'insieme di esempi (training set) per i quali è conosciuta la classe, i pattern sconosciuti sono assegnati alla classe a cui appartiene il pattern del training set più vicino. Se il numero di campioni è elevato si considerano i primi k pattern del trai-

Primi esperimenti

Per chi volesse già iniziare a cimentarsi con del codice consiglio caldamente di scaricarsi il PR-Tool 3.0 (Matlab), nei prossimi articoli gli esempi mostrati si baseranno sulle funzioni di questo Tool.

ning set (k-NN) e la classe presente fra questi con maggiore frequenza viene attribuita al pattern sconosciuto.

Approccio strutturale - In questo caso, per la rappresentazione dei pattern non si usano vettori numerici, ma si utilizzano strutture simboliche (stringhe, grafi..) per la descrizione delle relazioni (spaziali, temporali) fra i componenti elementari dei pattern. Si tratta di metodi nei quali non sono tanto importanti i valori assunti dalle feature, quanto le relazioni esistenti fra i vari componenti dei pattern. La classificazione di un pattern sconosciuto viene effettuata confrontando la sua rappresentazione con un insieme di predefiniti pattern modello o prototipi che rappresentano le varie classi. La struttura più semplice per la rappresentazione dei pattern è la stringa, una sequenza finita di simboli, ognuno dei quali rappresenta una componente *atomica* del pattern. Il confronto può avvenire semplicemente calcolando il costo delle trasformazioni per passare da una stringa all'altra, dove il costo è il numero di trasformazioni elementari effettuate (sostituzione, inserimento ed eliminazione di un simbolo), tale valore è detto distanza di Levensthein.

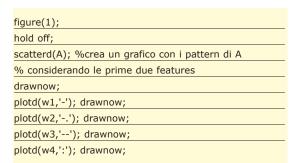
Reti Neurali - Infine, citiamo brevemente le reti neurali, già trattate nei vari articoli di ioProgrammo. Le reti neurali sono (in maniera semplicistica) un grafo in cui i nodi processano le informazioni trasmesse da altri nodi ad essi collegati. Il vero punto di forza di questi metodi è che possono codificare anche complessi fenomeni non lineari, che solitamente vengono appresi da esempi. Il fatto di considerare unicamente la classificazione sui pattern del training set porta al problema che spesso non si raggiunge una soluzione ottimale al problema (cadiamo in un minimo locale). Proprio per superare questo limite negli ultimi anni sono state sviluppate le Support Vector Machine. Le vedremo nei prossimi articoli .

INTRODUZIONE AL PRTOOL 3.0

Nel corso di questa breve serie di articoli sulla Pattern Recognition utilizzeremo una serie di librerie scaricabili da Internet, le *PatternRecognitionTool* (le trovate anche nel CD allegato alla ricista). Vediamo un primo piccolo esempio per capire il funzionamento di tale tool. Per prima cosa definiamo la classe (*classe* dal punto di vista della programmazione) *dataset*. Mediante questa memorizziamo il training (dati di esempio) / test (dati da classificare) set nel seguente modo: *dataset* (vettori del training, classi di appartenenza dei vettori del training). *N.B.* tutti i dati vengono gestiti dal Tool utilizzando la classe dataset.

```
A = gendatd(100,100,25);
[C,D] = gendat(A,80);
w1 = knnc(C,1);
w2 = knnc(C,3);
w3 = knnc(C,7);
w4 = knnc(C,11);
disp([testd(D*w1),testd(D*w2),testd(D*w3),testd(D*w4)]);
figure(1);
hold off;
scatterd(A); drawnow;
plotd(w1,'-'); drawnow;
plotd(w2,'-.'); drawnow;
plotd(w3,'--'); drawnow;
plotd(w4,':'); drawnow;
```

Cerchiamo di capire come funziona questo semplice programma. Il comando *gendatd()* non fa altro che creare un training set di dati aventi una distribuzione normale con sovrapposizione fra le due classi. I primi due 100 sono la cardinalità delle due classi, mentre 25 è il numero di feature. Il comando *gendat()* prende 80 elementi da ogni classe di A e crea il training set C, gli atri 20 elementi formano il test set D Il comando *knnc(C,k)* effettua un k-Nearest Neighbor, l'output di tale funzione è una classe(N.B. classe dal punto di vista della programmazione) di tipo mapping, il risultato di un mapping * dataset è ancora un dataset che contiene i risultati della classificazione del dato dataset usando il classificatore che ha creato il mapping. Il comando *testd()* semplicemente dà in output l'errore commesso nella classificazione. Infine le seguenti righe:

```
figure(1);
hold off;
scatterd(A); %crea un grafico con i pattern di A
% considerando le prime due features
drawnow;
plotd(w1,'-'); drawnow;
plotd(w2,'-.'); drawnow;
plotd(w3,'--'); drawnow;
plotd(w4,':'); drawnow;
```

creano il seguente grafico (Fig. 5). Nei prossimi articoli introduttivi continueremo la spiegazione del tool con esempi sempre più complessi.

Loris Nanni

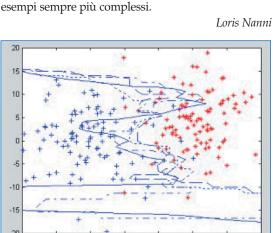



Fig. 5: **L'output grafico del tool. Le croci rosse e blu rappresentano le due classi, le linee tratteggiate sono le linee di separazione create dai classificatori.**

Pattern Recognition

Bibliografia

- INTRODUCTION ON STATISTICAL PATTERN RECOGNITION 2nd edition *K.Fukunaga* (Academic Press, Inc.) 1990

- PRTOOLS 3.0: A MATLAB TOOLBOX FOR PATTERN RECOGNITION, 2000 *R.Duin* (Delft University of Technology, Holland)

Soap

Web Services

TECNICHE DI ACCESSO A DATABASE REMOTI

seconda parte

Il mese scorso abbiamo visto come sia possibile costruire, in modo piuttosto semplice, un Web Service utilizzando la piattaforma Microsoft .Net ed in particolare il Microsoft .Net Framework SDK.

File sul Web
www.itportal.it\iop68
Soap.zip

Agenda
In questo articolo svilupperemo una serie di client in grado di consumare i vari servizi messi a disposizione.

Il mese scorso abbiamo visto come scrivere un servizio di profilatura remota implementato come Web Service utilizzando la tecnologia Microsoft .Net. Il servizio è stato realizzato in linguaggio C# attraverso il file *profiling.asmx*, pubblicato su Internet Information Services e direttamente utilizzabile attraverso un client HTTP oppure via SOAP. In prima battuta, il servizio è immediatamente disponibile alle chiamate HTTP effettuate attraverso il browser Internet Explorer all'url: *http://localhost/webservices/profiling.asmx*. Questa request mostra la pagina di descrizione del servizio visibile in Fig. 1.

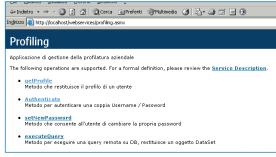



Fig. 1: Pagina di descrizione del servizio di profilatura.

In questo articolo svilupperemo una serie di client in grado di consumare i vari servizi messi a disposizione da questo Web Service di profilatura.

I SERVIZI

Il Web Service che abbiamo sviluppato espone quattro metodi:

```
public bool Authenticate(string Username, string Password)
```

riceve la coppia di stringhe *Username* e *Password* e restituisce il valore booleano *true*, se sul database localizzato sul server, esiste un record con quei valori nei campi *Username* e *Password*. In caso contrario, restituisce il valore *false*.

```
public DataSet getProfile(string Username, string Password)
```

riceve la coppia di stringhe *Username* e *Password* e restituisce un oggetto *DataSet*. Se sul database esiste un record con quei valori nei campi *Username* e *Password*, il *DataSet* conterrà il profilo dell'utente identificato da quel particolare *Username*, in caso contrario o in caso di errore, il *DataSet* conterrà un solo record e nel campo *Message* ci sarà il messaggio di errore.

```
public string setNewPassword(string Username, string
                    OldPassword, string NewPassword)
```

riceve le stringhe *Username, OldPassword* e *NewPassword* e se sul database esiste un record conforme alla coppia *Username / OldPassword* viene eseguita un'istruzione di *UPDATE* su quel record per modificarne l'informazione della password da *OldPassword* a *NewPassword*. Il risultato può essere la stringa *"ok"* se non ci sono stati problemi, la stringa *"invalid login"* se l'autenticazione è fallita oppure un messaggio di errore se qualcosa non ha funzionato (ad esempio nel caso in cui il database non sia raggiungibile).

```
public DataSet executeQuery(string Query)
```

riceve una stringa, la interpreta come query da eseguire sul database, la esegue e restituisce un oggetto *DataSet* contenente tutti i record selezionati, se si tratta di una query di aggiornamento il *DataSet*, sarà vuoto. In caso di errore, il *DataSet* conterrà il messaggio di errore occorso.

Come vediamo, si tratta di un ventaglio di opzioni piuttosto ampio, a questo punto non ci resta che scrivere la classe proxy ed i client che, attraverso questa, saranno in grado di utilizzare i servizi remoti messi a disposizione dal Web Service.

IL PROXY

Come sappiamo, è possibile, attraverso i tool che ci sono messi a disposizione dal .Net Framework SDK, creare in maniera automatica una classe che faccia da proxy verso i possibili consumatori del nostro servizio. Attraverso questo oggetto sarà possibile utilizzare il Web Service senza preoccuparci delle modalità con le quali questo viene realmente chiamato ed utilizzato. Una classe *proxy* è un oggetto .Net, utilizzato direttamente da un'applicazione terza, il cui compito è rendere trasparente al consumatore l'accesso al Web Service. Per creare automaticamente il proxy, si utilizza il tool wsdl.exe fornito con il .Net Framework SDK, il comando da eseguire dal prompt dei comandi è il seguente:

```
wsdl /l:cs /n:max.profiling http://localhost/
                              WebServices/profiling.asmx?WSDL
```

In questo caso, si chiede al tool *wsdl.exe* di generare, in linguaggio C#, una classe *proxy* verso il Web Service il cui file di descrizione WSDL sia quello indicato e raggiungibile via HTTP. Il *NameSpace* di riferimento da utilizzare per la generazione del nostro assembly, sarà quello indicato all'opzione */n*, cioè *max.profiling*. Il risultato dell'esecuzione di questo comando sarà il seguente:

```
Microsoft (R) Web Services Description Language Utility
[Microsoft (R) .NET Framework, Version 1.0.3705.0]
Copyright (C) Microsoft Corporation 1998-2001.
                                         All rights reserved.
Writing file 'C:\Inetpub\wwwroot\WebServices \Profiling.cs'.
```

Verrà quindi generato, nella directory locale, il file *Profiling.cs*, cioè il sorgente C# dell'assembly che costituisce la classe proxy per l'accesso al servizio.
A questo punto, non ci resta che compilarlo attraverso il comando:

```
csc /t:library /out:profiling.dll profiling.cs /r:system.dll
                    /r:system.xml.dll /r:system.web.services.dll
```

Stiamo sfruttando una caratteristica del compilatore C# *csc.exe*, fornito a corredo con il .Net Framework SDK, che ci permette di generare un assembly da utilizzare come libreria per altre componenti software, questo avviene attraverso l'utilizzo dell'opzione */t:library*. L'effetto dell'esecuzione del comando, sarà la compilazione del sorgente C# e la produzione dell'assembly *profiling.dll*. A questo punto, possediamo un assembly in grado di svolgere funzionalità di proxy per tutte le applicazioni che vorranno accedere ai servizi messi a disposizione del nostro Web Service.

APPLICAZIONE DI AUTENTICAZIONE

Sfruttiamo subito il nostro proxy scrivendo un programma C# in grado di eseguire una semplice operazione di login sul sistema remoto e di restituire un messaggio che ci indichi se l'autenticazione è andata a buon fine. Ecco il sorgente del file login.cs:

```
using System;
using System.Data;
using max.profiling;
public class Login
{
  public static void Main()
  {
    Profiling myProfile = new Profiling();
    Console.Write("Username ?: ");
    string u = Console.ReadLine();
    Console.Write("Password ?: ");
    string p = Console.ReadLine();
    bool login_ok = myProfile.Authenticate(u,p);
    if (login_ok)
       Console.Write("Login OK");
    else
    Console.Write("Wrong username or password");
  }
}
```

Analizzando il sorgente, appare chiaro che l'applicazione non utilizza direttamente il Web Service, ma utilizza più semplicemente un oggetto *myProfile*, istanza della classe *Profiling* (il proxy che abbiamo recentemente costruito) e questo è perfettamente legittimo in quanto, tra i namespace importati, c'è anche l'istruzione *using max.profiling*; che è esattamente il namespace della nostra classe proxy. Una volta istanziato l'oggetto, sarà sufficiente utilizzare i suoi metodi che, guarda caso, saranno esattamente quelli definiti a livello di Web Service e che erano stati esportati attraverso il file WSDL di descrizione del servizio. Il file WSDL era stato infatti utilizzato per produrre il proxy che adesso stiamo utilizzando. Pertanto l'istruzione bool *login_ok = myProfile.Authenticate(u,p);* provoca l'esecuzione del metodo remoto a cui vengono passati i parametri di username e password e che restituisce un valore booleano che indica l'effettiva validità delle credenziali per l'autenticazione. Questo sorgente C# deve essere compilato attraverso il comando:

```
csc login.cs /r:profiling.dll
```

Da notare, in questo caso, l'utilizzo del parametro /r per indicare al compilatore che, tra i vari assembly da utilizzare per risolvere i namespace, c'è anche il nostro proxy verso il Web Service.
L'utilizzo della funzione di autenticazione, visibile in Fig. 2, è piuttosto semplice, una volta lanciata l'applicazione, questa ci chiede di inserire uno username ed una password. Dopo aver effettuato la verifica della validità della coppia, verifica che è a carico del Web Service, l'applicazione restituisce una stringa

Soap

Classe proxy

Una classe proxy è un oggetto .Net, utilizzato direttamente da un'applicazione terza, il cui compito è rendere trasparente al consumatore l'accesso al Web Service. Per creare automaticamente il proxy, si utilizza il tool *wsdl.exe* fornito con il .Net Framework SDK.

Una questione di sicurezza...

Cosa pensereste, se veniste a sapere che la password del vostro account di posta privata o del vostro conto online è gestita con la massima riservatezza ed attenzione durante il trasporto ma viene memorizzata in una tabella sulla quale è sufficiente fare una query per ottenerla stampata a video? Diffidate quindi di quei servizi che in caso di smarrimento vi rispediscono la vostra vecchia password.

```
Prompt dei comandi

C:\Inetpub\wwwroot\WebServices>login
Username ?: u0e12345
Password ?: Goldrake
Login OK
C:\Inetpub\wwwroot\WebServices>login
Username ?: u0e12345
Password ?: pippo
Wrong username or password
C:\Inetpub\wwwroot\WebServices>
```

Fig. 2: **Utilizzo della funzione di autenticazione.**

che indica la validità o meno delle informazioni che abbiamo inserito.

APPLICAZIONE DI GESTIONE PROFILI

Adesso ci apprestiamo ad analizzare un'applicazione leggermente più complessa, in grado di utilizzare il metodo *getProfile()* esposto dal Web Service e, di conseguenza, in grado di trattare l'oggetto di tipo *DataSet* che il metodo remoto restituisce al chiamante. Si tratta di un client che passa username e password al servizio remoto e, se l'autenticazione è andata a buon fine, stampa a video il completo profilo di sicurezza dell'utente. Ecco il sorgente del file profile.cs:

```
using System;
using System.Data;
using max.profiling;
public class Profile
{ public static void Main()
  { Profiling myProfile = new Profiling();
    Console.Write("Username ?: ");
    string u = Console.ReadLine();
    Console.Write("Password ?: ");
    string p = Console.ReadLine();
    DataSet myDS = myProfile.getProfile(u,p);
    foreach(DataRow myRow in myDS.Tables[0].Rows)
    { Console.WriteLine();
      foreach (DataColumn myColumn in
                                    myDS.Tables[0].Columns)
      {
        Console.WriteLine(myRow[myColumn]);
      }
    }
  }
}
```

In questo sorgente c'è da notare che, comunque sia implementato il nostro servizio remoto, ci aspettiamo che il valore restituito sia esattamente di tipo *DataSet*. L'applicazione, infatti, andrà a scrivere sulla console il valore di tutte le colonne di tutte le righe del *DataSet myDS*, quindi otterremmo un errore di runtime se il servizio restituisse un oggetto di tipo differente. In ogni caso, la nostra applicazione non può correre questo rischio in quanto utilizza un proxy che è stato costruito a partire proprio dalla descrizione dinamica del servizio, pertanto non ci sono dubbi sul tipo dell'oggetto restituito dal metodo remoto. Il funzionamento di questo metodo remoto, è basato sul fatto che il tipo restituito possa essere serializzato in XML e trasmesso via HTTP dal server verso il client. Sappiamo che questo è esattamente quello che accade, infatti chiamando direttamente il servizio remoto attraverso il browser puntato all'url seguente: *http://localhost/webservices/profiling.asmx/getProfile?Username=u0e12345&Password=Goldrake* otterremo quanto mostrato in Fig. 3 e questo grazie all'utilizzo di un particolare namespace.

```
http://localhost/webservices/profiling.asmx/getProfile?Username=u0e12345&Password=Goldrake - Microsoft Interne
File Modifica Visualizza Preferiti Strumenti ?
Indietro Cerca Preferiti Multimedia
Indirizzo http://localhost/webservices/profiling.asmx/getProfile?Username=u0e12345&Password=Goldrake

  <?xml version="1.0" encoding="utf-8" ?>
- <DataSet xmlns="max.profiling">
  - <xs:schema id="NewDataSet" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
    - <xs:element name="NewDataSet" msdata:IsDataSet="true" msdata:Locale="it-IT">
      - <xs:complexType>
        - <xs:choice maxOccurs="unbounded">
          - <xs:element name="Utenti">
            - <xs:complexType>
              - <xs:sequence>
                  <xs:element name="ID" type="xs:int" minOccurs="0" />
                  <xs:element name="Username" type="xs:string" minOccurs="0" />
                  <xs:element name="Password" type="xs:string" minOccurs="0" />
                  <xs:element name="Nome" type="xs:string" minOccurs="0" />
                  <xs:element name="Cognome" type="xs:string" minOccurs="0" />
                  <xs:element name="Indirizzo" type="xs:string" minOccurs="0" />
                  <xs:element name="Email" type="xs:string" minOccurs="0" />
                  <xs:element name="Struttura" type="xs:string" minOccurs="0" />
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:choice>
        </xs:complexType>
Operazione completata                                    Intranet locale
```

Fig. 3: **La risposta XML del server.**

Se proviamo a guardare il payload XML trasmesso dal server verso l'applicazione client noteremo infatti, l'utilizzo del namespace *xmlns:msdata="urn:schemas-microsoft-com: xml-msdata"* in grado, in generale, di descrivere le politiche di serializzazione di molti dei tipi complessi utilizzabili dai linguaggi supportati dalla piattaforma Microsoft .Net. Anche in questo caso, il codice C# deve essere compilato attraverso il comando:

```
csc profile.cs /r:profiling.dll
```

L'utilizzo dell'applicazione di gestione del profilo utente è visibile in Fig. 4.

```
Prompt dei comandi

C:\Inetpub\wwwroot\WebServices>profile
Username ?: u0e12345
Password ?: Goldrake

1
u0e12345
Goldrake
Marcello
Rossi
Via dei mandorli 192
rossi@ilmiosito.it
Amministrazione

C:\Inetpub\wwwroot\WebServices>profile
Username ?: Goldrake
Password ?: Actarus

Wrong Username or Password

C:\Inetpub\wwwroot\WebServices>
```

Fig. 4: **Utilizzo dell'applicazione di gestione del profilo utente.**

Da notare, in questo caso, l'utilizzo di un database all'interno del quale le password degli utenti sono memorizzate in chiaro. Questa, anche se comoda per i nostri esempi, è una politica che deve essere sempre evitata per motivi di sicurezza e di riservatezza verso un dato particolarmente sensibile che l'utente finale affida alle nostre applicazioni.

APPLICAZIONE DI MODIFICA DELLA PASSWORD

In qualsiasi sottosistema di profilatura utente non può mancare una funzionalità di modifica della password. Il Web Service che utilizziamo espone un metodo che fa al caso nostro, cerchiamo di utilizzarlo scrivendo un'applicazione ad hoc. Ecco il sorgente del file changepwd.cs:

```
using System;
using System.Data;
using max.profiling;
public class Changepwd
{ public static void Main()
  { Profiling myProfile = new Profiling();
    Console.Write("Username ?: ");
    string u = Console.ReadLine();
    Console.Write("Old password ?: ");
    string op = Console.ReadLine();
    Console.Write("New password ?: ");
    string np = Console.ReadLine();
    Console.WriteLine(myProfile.setNewPassword
                                        (u,op,np))
  }
}
```

Questa piccola applicazione non è altro che un'interfaccia di console verso il Web Service. Riceve infatti i parametri necessari a compiere l'operazione di modifica della password dell'utente e, dopo aver invocato il servizio remoto, non fa altro che restituire sulla console il messaggio ottenuto dal servizio stesso. Anche in questo caso, il codice C# deve essere compilato attraverso il comando:

```
csc changepwd.cs /r:profiling.dll
```

L'utilizzo dell'applicazione di modifica della password è visibile in Fig. 5. Si vede in particolare come, dopo aver modificato la password di un utente, non sia ovviamente più possibile effettuare l'autenticazione con le vecchie credenziali, ma sia invece immediatamente possibile farlo con le nuove. Anche se in linea generale ci interessa soltanto l'interfaccia del servizio, cioè cosa fa e non come lo fa, in questo caso assume una qualche importanza il fatto che, all'interno del metodo remoto che si occupa di restituire al chiamante il profilo dell'utente, la procedura di autenticazione sia stata demandata ad una funzione privata e generalizzata all'interno del servizio stesso.

```
Prompt dei comandi

C:\Inetpub\wwwroot\WebServices>changepwd
Username ?: u0e12345
Old password ?: Goldrake
New password ?: Jeeg
ok

C:\Inetpub\wwwroot\WebServices>login
Username ?: u0e12345
Password ?: Goldrake
Wrong username or password
C:\Inetpub\wwwroot\WebServices>login
Username ?: u0e12345
Password ?: Jeeg
Login OK
C:\Inetpub\wwwroot\WebServices>changepwd
Username ?: u0e12345
Old password ?: Goldrake
New password ?: Jeeg
invalid login

C:\Inetpub\wwwroot\WebServices>changepwd
Username ?: u0e12345
Old password ?: Jeeg
New password ?: Goldrake
ok

C:\Inetpub\wwwroot\WebServices>
```

Fig. 5: **Utilizzo dell'applicazione di modifica della password.**

Si tratta, in particolare, della funzione:

```
private bool isValidLogin(string Username,
                              string Password) {...}
```

che essendo stata dichiarata private è accessibile soltanto ai metodi presenti all'interno della classe stessa e che si occupa della funzionalità elementare dell'autenticazione. Per quanto riguarda l'implementazione del client che permetta la modifica della password, c'è da notare che manca totalmente la gestione degli errori lato client. Mentre gli errori lato server, per esempio l'assenza di connettività del database, sono a carico del servizio, nel caso di errori lato client, per esempio irraggiungibilità del Web Service, l'applicazione sarebbe totalmente impreparata nel gestirli e fallirebbe miseramente. Chi ha scritto il consumatore del servizio, ahimè, farebbe una figura da principiante. Questi aspetti sono irrilevanti nel contesto di questo articolo, pertanto sono stati trascurati, ma in contesti di produzione è fondamentale tenerne conto.

QUERY SU DATABASE REMOTO

L'ultimo servizio esposto dal Web Service che stiamo utilizzando, ci permette di eseguire una qualsiasi query sul database remoto, scriviamo a questo punto un client che ci permetta di utilizzarlo. Si tratta del file *query.cs* ed ecco il suo sorgente:

```
using System;
using System.Data;
using max.profiling;
public class Query
{ public static void Main()
  { Profiling myProfile = new Profiling();
    Console.Write("Query ?: ");
    string q = Console.ReadLine();
```

In sintesi

In questo articolo abbiamo visto come si possa utilizzare il Web Service di gestione dei profili utente sviluppato nell'articolo del mese scorso. In particolare abbiamo realizzato, utilizzando il linguaggio C#, la classe proxy per l'accesso remoto al servizio e quattro consumatori, uno per ogni metodo remoto che il servizio mette a disposizione. Abbiamo visto come sia possibile ottenere il trasporto dal Web Service al consumatore di oggetti semplici (tipi primitivi) come di oggetti complessi (interi *DataSet*) senza dover alterare i comportamenti né del servizio né dei client.

Soap

```
    DataSet myDS = myProfile.executeQuery(q);
    if (myDS.Tables.Count != 0)
      foreach(DataRow myRow in myDS.Tables[0].Rows)
      { Console.WriteLine();
        foreach (DataColumn myColumn in
                             myDS.Tables[0].Columns)
        {Console.WriteLine(myRow[myColumn]);}
      } }
}
```

```
Prompt dei comandi

C:\Inetpub\wwwroot\WebServices>query
Query ?: select * from utenti

1
u0e12345
Goldrake
Marcello
Rossi
Via dei mandorli 192
rossi@ilmiosito.it
Amministrazione

2
u0e33213
bisonte
Fausto
Bianchi
Via dei sicomori 133
bianchi@iltuosito.com
Personale

3
u0e11223
casdckjflkejbfekj
Maria
Furbetta
Corso del tempo 1
maria@abracadabra.it
Uffici periferici

C:\Inetpub\wwwroot\WebServices>
```

Fig. 6: Utilizzo della query di selezione.

Come era accaduto in precedenza, siamo di fronte ad un client che richiede una stringa, la passa al metodo remoto e da questo si aspetta di ricevere un oggetto *DataSet* completo da poter poi stampare un pezzo alla volta sulla console. Il servizio, dal canto suo, non fa altro che ricevere la stringa che rappresenta la query, eseguirla sul database (remoto dal punto di vista del client, ma locale e direttamente raggiungibile dal punto di vista del servizio), riceverne un oggetto istanza della classe *DataSet* e restituirla al chiamante. Lo strato di gestione del Web Service si occuperà di serializzare l'oggetto da trasportare verso il client e, come accadeva per l'applicazione di gestione profili, si occuperà anche di deserializzare l'oggetto prima di passarlo al consumatore, in modo da rendere trasparente il fatto che l'accesso avvenga in modalità remota ad un Web Service piuttosto che localmente. Questa seconda fase viene svolta proprio grazie all'oggetto proxy che abbiamo creato in precedenza.
Per compilare il client si utilizza, come al solito, il seguente comando:

```
csc query.cs /r:profiling.dll
```

Per prima cosa proviamo ad effettuare una query di tipo selettivo, in particolare possiamo provare a selezionare tutti i campi di tutti i record, senza distinzione alcuna. In questo caso, come sappiamo, la query che dovremo passare al client sarà la seguente: select * from utenti. Il risultato di questa elaborazione, se non ci sono stati errori, è visibile in Fig. 6.
Come si può vedere il client ha ricevuto dal server un payload contenente tutto il DataSet che ci aspettavamo, cioè tutti i campi di tutti i record della tabella "utenti". Questo pacchetto di dati è stato poi stampato dal client sulla console. Naturalmente, avendo il massimo controllo sulla query che verrà eseguita in remoto, potremmo utilizzare il client anche in maniera più sofisticata.
Proviamo ad esempio a passare al client la query: *select * from utenti where id=1* ed otterremo il seguente risultato:

```
1
u0e12345
Goldrake
Marcello
Rossi
Via dei mandorli 192
rossi@ilmiosito.it
Amministrazione
```

Un risultato analogo, naturalmente, si ottiene se si utilizza la query:

```
select * from utenti where username='u0e12345'.
```

Proviamo a questo punto l'esecuzione di una query di aggiornamento passando al client la stringa: *UPDATE Utenti SET Utenti.Indirizzo = 'Via delle nuove vie' where Utenti.Username = 'u0e12345'*. Quello che ci aspettiamo, ed è esattamente quello che accade, è che dopo l'esecuzione il contenuto del campo Indirizzo del record identificato da quel particolare utente sia stato variato.
Naturalmente è così e per verificarlo non dovremo far altro che utilizzare la stessa query di selezione vista in precedenza.

```
Prompt dei comandi

C:\Inetpub\wwwroot\WebServices>query
Query ?: select * from utenti where username='u0e12345'

1
u0e12345
Goldrake
Marcello
Rossi
Via dei mandorli 192
rossi@ilmiosito.it
Amministrazione

C:\Inetpub\wwwroot\WebServices>query
Query ?: UPDATE Utenti SET Utenti.Indirizzo = 'Via delle nuove vie' where Utenti.Username
'u0e12345'

C:\Inetpub\wwwroot\WebServices>query
Query ?: select * from utenti where username='u0e12345'

1
u0e12345
Goldrake
Marcello
Rossi
Via delle nuove vie
rossi@ilmiosito.it
Amministrazione

C:\Inetpub\wwwroot\WebServices>_
```

Fig. 7: Esecuzione di una query di aggiornamento.

Il risultato di quanto detto è visibile in Fig. 7.

Massimo Canducci

Servizi remoti

Utilizzando opportunamente il servizio remoto ed il suo consumatore, è stato possibile accedere ad un database remoto effettuando query di selezione o query di aggiornamento, il tutto senza preoccuparci minimamente delle problematiche di serializzazione e di trasporto dell'informazione.

e-government

CRITTOGRAFIA E FIRMA DIGITALE

L' e-government, è l'insieme delle nuove iniziative che consentono di rispondere adeguatamente, efficientemente e tempestivamente, alla domanda di governo da parte degli attori sociali ed economici che cooperano e competono sulla scena mondiale.

Difendere la privacy individuale in un era di crescente informatizzazione sta diventando un problema di cruciale importanza con il quale tutti prima o poi dovremmo confrontaci. L'unica possibile via per difendere il proprio diritto alla privacy è adoperare la crittografia. Le applicazioni basate sulla crittografia si integrano efficacemente nelle tecnologie telematiche inerenti l'e-government.

Le principali esigenze avvertite in una transazione informatica, oltre alla riservatezza della comunicazione, sono:

- la certezza dell'identità dei vari interlocutori;
- la certezza che la documentazione scambiata tra i vari attori che partecipano alla transazione sia integra (cioè non sia stata modificata da eventuali malintenzionati infiltratisi nella comunicazione).

Queste prerogative vengono entrambe soddisfatte dall'applicazione della firma digitale ai documenti elettronici scambiati durante una transazione. In queste pagine implementeremo un sistema che permette di apporre la propria firma (digitale) su un documento informatico, di verificare che un documento firmato digitalmente non sia stato modificato, ed infine, che la firma apposta sul documento inviato da un certo mittente, sia autentica. L'applicazione sviluppata, utilizza i concetti e gli standard vigenti in materia di "FIRMA DIGITALE". Vengono tralasciati alcuni aspetti legali che limitano l'ambito di applicazione del prodotto (realizzato nella categoria di firma digitale "debole"), in cui non sono previste terze parti autorizzate che garantiscono sull'identità dei soggetti.

Tuttavia l'identità dei vari interlocutori viene curata attraverso un processo di autocertificazione, in cui ogni attore certifica la propria identità.

TECNICHE DI CRITTOGRAFIA

Al giorno d'oggi, la parola *crittografia* è usata per indicare una grande varietà di tecniche, il cui obiettivo congiunto è quello di garantire la completa riservatezza delle informazioni, consentendo applicazioni quali l'autenticazione, il denaro elettronico e molte altre ancora. La regola fondamentale su cui si basa la crittografia è la conoscenza, da parte del crittoanalista, del metodo di cifratura impiegato. La quantità di sforzi necessari per inventare, collaudare e installare un nuovo metodo, ogni volta che quello vecchio è compromesso (o si pensa che lo sia), ha sempre reso poco pratico il mantenimento di tale segreto. Entra quindi in gioco la chiave, che è una stringa di caratteri che seleziona una tra le molte cifrature potenziali. Tutti i moderni metodi utilizzano una chiave per eseguire la cifratura e la decifratura; un messaggio può essere decrittografato solo se la chiave di decifratura si "accoppia" con quella di cifratura. Per alcuni algoritmi le due chiavi sono uguali, mentre per altri esse sono diverse. In base a questa sostanziale differenza, le tecniche di cifratura basate sull' utilizzo di chiavi, si dividono in cifratura simmetrica (detta anche a chiave simmetrica o a chiave segreta) e asimmetri-

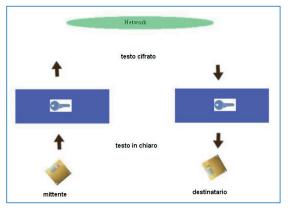



Fig. 1: **Cifratura simmetrica.**

Certification Autority

Soggetto pubblico o privato che effettua la certificazione, rilascia il certificato della chiave pubblica, lo pubblica unitamente a quest'ultima, pubblica ed aggiorna gli elenchi dei certificati sospesi e revocati.

AIPA

(Autorità per l'informatica nella pubblica amministrazione) ha il compito di approvare l'inserimento nell'elenco di un nuovo certificatore e di tenere aggiornato l'elenco stesso.

firma digitale

ca (detta anche a chiave asimmetrica o a chiave pubblica). In un sistema di *cifratura simmetrica* viene usata una sola chiave, detta appunto segreta, come parametro di una funzione unidirezionale e invertibile, permettendo così di elaborare il testo del messaggio da trasmettere in modo da renderlo incomprensibile agli intercettatori. Essendo la funzione invertibile, il destinatario dovrà soltanto elaborare nuovamente il crittogramma richiamando l'inversa della funzione di cifratura, avente come parametro la stessa chiave utilizzata dal trasmettitore del messaggio. Ovviamente la tecnica si basa sulla capacità del mittente e del destinatario di mantenere segreto il codice di cifratura. Tale metodo, noto da secoli, è definito crittografia simmetrica. Con il sistema a secret-key il mittente e il destinatario devono raggiungere un accordo sulla scelta della chiave. La cosa non sarà facile se essi si trovano a migliaia di chilometri di distanza e sicuramente questo è il caso più frequente, altrimenti perché farebbero uso di comunicazione telematica?
Come potranno allora scambiarsi la chiave?
Questi problemi sono risolti dalla crittografia a chiave pubblica.
Le *tecniche asimmetriche*, utilizzano coppie di chiavi complementari invece di una sola chiave segreta. Un singolo utente possiede una coppia univoca di chiavi complementari; di queste, una è una chiave pubblica, nel senso che può essere conosciuta da tutti, ed è usata per cifrare il messaggio, mentre l'altra è una chiave privata ed è tenuta al sicuro dal suo proprietario di modo che solo lui possa utilizzarla. Le due chiavi sono create in maniera tale che un messaggio cifrato da una delle due può essere decifrato solo e soltanto dall'altra. In pratica, se si vuole spedire un messaggio a una certa persona, si codifica quel messaggio utilizzando la sua chiave pubblica, e si è sicuri che soltanto quella persona potrà decifrarla con la propria chiave privata: neanche la chiave pubblica utilizzata per crittografare il messaggio riuscirà a decrittografare il messaggio stesso.
Oltre alla cifratura di un messaggio, le chiavi asimmetriche possono essere utilizzate per generare la "firma digitale" di un qualsiasi documento informatico.

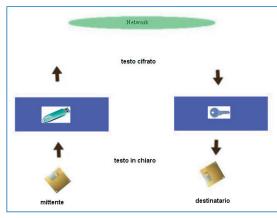



Fig. 2: Cifratura asimmetrica

LA FIRMA DIGITALE: CHIAVE E CERTIFICATO

Il processo di firma si basa sull'impiego di una coppia di chiavi asimmetriche: la chiave privata, impiegata dal mittente per firmare il documento, e la chiave pubblica, contenuta in un certificato, usata dal destinatario per verificare l'integrità dello stesso. L'utente che vuole usare il sistema di firma si deve munire delle sopraccitate chiavi. Il processo di generazione delle chiavi, si basa sull'ausilio di particolari algoritmi matematici, progettati in maniera tale da garantire l'assoluta casualità del processo di generazione, l'impossibilità di ricavare la chiave privata da quella pubblica e viceversa. Gli algoritmi impiegati in questo processo sono noti in letteratura come:

- RSA (Rivest-Shamir-Adleman algorithm)
- DSA (Digital Signature Algorithm)

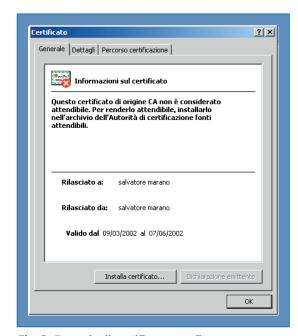



Fig. 3: Esempio di certificato autofirmato

Tali algoritmi sono esattamente quelli impiegati per la generazione delle chiavi nella cosiddetta firma certificata o forte, e sono supportati dalle leggi vigenti in materia. Dopo la generazione delle chiavi, bisogna certificare la chiave pubblica; il processo di certificazione ha lo scopo di rassicurare chiunque riceva un documento correttamente firmato, circa l'identità del soggetto che ha apposto la firma. La mancanza di Certification Autority (CA) non accreditata AIPA relega l'applicazione nella categoria "firma debole". Il processo di certificazione si riduce alla creazione di un certificato digitale, che con-

Firma digitale

Al seguito della diffusione dell'e-commerce, è sorto il problema di garantire la sicurezza delle transazioni in Rete. La firma digitale è nata proprio da questa esigenza. Si tratta di un sistema di crittografia che, attraverso l'applicazione di un opportuno codice, consente l'accesso al documento soltanto a coloro che hanno la chiave di decifrazione. In tal modo, vengono garantite l'integrità del documento (che potrà essere modificato esclusivamente da colui che lo ha creato) e la sua riservatezza (può leggerlo solo il destinatario).

X509

Standard che definisce i formati e le informazioni che un certificato digitale deve contenere.

tiene la chiave pubblica insieme ad altre informazioni, riguardanti l'identità del richiedente, secondo le direttive dettate in materia dallo standard X509 v2. La certezza dell'identità del soggetto la otteniamo mediante una sorta di autocertificazione, ovvero mediante la firma del certificato con la chiave privata del soggetto stesso. Il certificato mostrato nella Fig. 1 non è considerato attendibile o affidabile poiché questo non è stato rilasciato da una CA riconosciuta dall'AIPA, come ad esempio Verisign o Infocamere ecc. La chiave privata e il corrispondente certificato autofirmato vengono memorizzati in un archivio, chiamato *keystore*, il quale è stato implementato attraverso un file che risiede nella home directory della macchina deputata al processo di creazione delle chiavi. I certificati vengono identificati all'interno del keystore attraverso un nome, noto come alias. Le chiavi private vengono associate univocamente a un certificato; l'accesso a queste ultime, all'interno dell'archivio, è protetto da password che non sono criptate. La piattaforma Java mette a disposizione, attraverso il framework denominato Java Cryptography Architecture (JCA), una serie di strumenti per il supporto della firma digitale e dei servizi di crittografia più comunemente utilizzati. Poiché nessuna delle classi dello standard JDK supporta la generazione dei certificati, ma solo la loro lettura, per la generazione e la gestione di certificati occorre introdurre i concetti di keystore e keytool. Un keystore è un insieme di chiavi e di certificati. Solitamente esso viene memorizzato in un file, ma può anche essere memorizzato in un altro posto, come un database o un server LDAP. Le chiavi e i certificati, vengono memorizzati come voci di un keystore, e si fa riferimento ad esse attraverso un nome, noto come alias. In un keystore ci sono quindi due tipi diversi di voci: i certificati e le chiavi. Il JDK utilizza i certificati X509, che è lo standard maggiormente usato per i certificati.

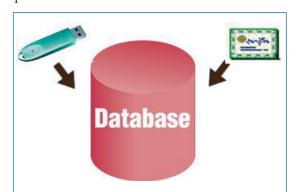



Fig. 4: **Chiave privata e il certificato vengono conservati in un keystore.**

Nell'applicazione in particolare si sono usati certificati X509 v2. Le classi che consentono la manipolazione di questi certificati sono localizzate nel package java.security.cert e sono:

- Certificate
- X509Certificate
- CertificateFactory

Le chiavi, invece, sono chiavi private che sono utlizzate per la firma dei documenti. Una chiave del keystore è associata a un certificato per quella chiave. Per controllare l'accesso alle chiavi i keystore ricorrono a password che non sono cifrate. La piattaforma Java mette a disposizione un keystore predefinito che nell'applicazione è usato per conservare le chiavi e i certificati. Questo keystore è memorizzato come un file con estensione, .keystore, e risiede nella home directory della macchina deputata alla gestione delle chiavi. *Keytool* è un'applicazione, fornita con il JDK, che permette la generazione di chiavi private e dei corrispondenti certificati autofirmati, che vengono realizzati secondo lo standard X509 v2. La lista completa delle operazioni che è possibile implementare con il keytool è consultabile sulla documentazione JAVA. Per generare un certificato usando l'algoritmo predefinito, DSA, basta eseguire il seguente comando dal prompt di MS-DOS.

```
C:\>keytool -genkey -alias marano
```

Per cambiare l'algoritmo da DSA in RSA è sufficiente usare lo switch -keyalg. Ad esempio l'opzione "-keyalg RSA" consentirà la generazione di un certificato con le chiavi RSA. Keytool aprirà il keystore predefinito nella home directory. Per specificare il nome del keystore è sufficiente usare il comando keystore[filename]. Verrà quindi chiesta una password del keystore. Se il keystore non è stato mai utilizzato prima, è sufficiente prendere una password, che keytool imposterà in modo che sia la password per questo keystore. Dopo l'inserimento della password, verranno formulate alcune domande inerenti il soggetto che richiede il certificato come è raffigurato nella Fig. 5.

Fig. 5: **Processo di generazione delle chiavi.**

Prima di generare il certificato, keytool chiede di confermare i dati, quindi provvederà alla sua generazione. L'applicazione, inoltre, chiederà una password necessaria per proteggere l'accesso alla chiave privata. A questo punto nel keystore è stata in-

Keystore e Keytool

Un Keystore è un insieme di chiavi e certificati. Ogni chiave è associata ad un certificato. Per accedere al keystore bisogna avere una password. Java mette a disposizione un keystore predefinito ed un keytool. Quest'ultimo consente di generare le chiavi private ed i corrispondenti certificati autofirmati.

Database

Insieme organizzato di dati utilizzati per il supporto allo svolgimento delle attività di un ente (azienda, ufficio, persona).

firma digitale

serita una chiave privata e il relativo certificato, la prima identificata utilizzando la password e il secondo attraverso un alias. Per ovvi motivi di sicurezza, la chiave privata viene consegnata direttamente al richiedente e memorizzata su un supporto magnetico che nel nostro caso è un floppy disk. Per estrarre la chiave privata dal keystore, si utilizza un'applicazione sviluppata in JAVA, che, ricevendo in input la password relativa al keystore e alla chiave privata da estrarre nonché all'alias del relativo certificato, effettua un'interrogazione al keystore estraendo la chiave selezionata. Per aumentare il livello di sicurezza, prima di memorizzare la chiave nel file da consegnare al richiedente, l'applicazione provvede a codificare la stessa . Il file che contiene la chiave privata codificata assumerà il nome del certificato con l'aggiunta dell'estensione .pk. Relativamente al precedente esempio, la chiave privata sarà contenuta in un file chiamato marano.pk, il quale sarà consegnato direttamente al richiedente, memorizzato su un floppy disk. Vediamo come avviene il dialogo tra l'applicazione e il keystore usando la classe java.security.KeyStore. Poiché il keystore si trova nella home directory della macchina del gestore, bisogna costruire il percorso per individuarlo:

```
String userHome=System.getProperty("user.home");
String keystoreFilename=userHome + File.separator
                                    + ".keystore";
```

Individuato il keystore, bisogna passare gli argomenti necessari alla selezione della chiave:

```
String alias=aliasCertificato;
char[] password=keystorePassword.toCharArray();
char[] KeyPassword=privateKeyPassword.toCharArray();
```

Si aprirà ora il file del keystore e si otterrà una istanza di KeyStore per gestire i dati di quel file.

```
FileInputStream fIn= new FileInputStream(
                              keystoreFilename);
  KeyStore keystore=KeyStore.getInstance("JKS");
  BufferedInputStream ksbufin= new
                         BufferedInputStream(fIn);
```

Per caricare il keystore, il flusso di input e la relativa password vengono passati in una istanza di KeyStore.

```
keystore.load(ksbufin,password);
```

Quindi è possibile recuperare la chiave privata, codificarla e salvarla in un file.

```
PrivateKey priv=(PrivateKey) keystore.getKey(alias ,
                                    KeyPassword);
byte[] KeyPriv = priv.getEncoded();
```

```
FileOutputStream keyPrivatefos =
              new FileOutputStream(nomeDellaChiave);
keyPrivatefos.write( KeyPriv);
keyPrivatefos.close();
```

Assemblando tutte le parti precedentemente descritte, otteniamo la classe che consente di selezionare ed esportare la chiave privata all'interno del keystore. Unitamente alla chiave di firma, al richiedente viene consegnata una copia del relativo certificato di verifica. Bisogna effettuare quindi l'estrazione dello stesso dal keystore e memorizzarlo in un file la cui estensione, compatibile con lo standard X509, sia .cer o .crt. L'operazione viene effettuata usando il comando:

```
C:\ >keytool -export -alias marano -file marano.cer
```

Il keytool provvederà all'estrazione del certificato identificato dall'alias e lo memorizzerà in un file la cui estensione è .cer.

LA FIRMA

La nostra architettura di riferimento (Fig. 6), è caratterizzata da un messaggio e/o documento, che deve viaggiare attraverso la rete tra due soggetti (Mittente e Destinatario). Il problema è quello di garantire l'autenticità delle informazioni trasmesse e la loro integrità; quindi una qualsiasi manomissione del messaggio trasmesso deve poter essere rilevata dal destinatario. Per soddisfare tale esigenza è necessario che, per ogni soggetto in grado di inviare messaggi, siano generate due chiavi asimmetriche : una pubblica, attraverso la quale coloro che riceveranno i messaggi effettueranno la verifica, e una privata con la quale sarà firmato il messaggio da trasmettere. A questo punto per la generazione della firma sarà necessario dotarsi di un algoritmo di cifratura: quello impiegato nell'applicazione realizzata è il DSA. Solitamente questi algoritmi non vengono applicati direttamente sul mes-

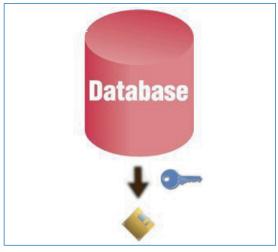



Fig. 6: Consegna della chiave privata.

Firma forte e Firma debole

Un documento caratterizzato da "firma forte" è equivalente ad un certificato cartaceo con firma autografa. Accanto a questo tipo di sottoscrizione elettronica, troviamo la cosiddetta "firma debole". Essa assicura solo la provenienza del documento, ma non l'integrità del contenuto. In pratica, la "firma forte" resta l'unica firma che attribuisce al documento informatico una valenza probatoria.

ServerLDAP

Il protocollo LDAP (Lightweight Directory Access Protocol) è uno standard aperto per i servizi su una rete Intranet o Internet. Una directory gestita dal protocollo LDAP è simile a una guida telefonica e può gestire molte altre informazioni, ma allo stato attuale viene usato principalmente per associare nomi a numeri telefonici e a indirizzi e-mail.

saggio, ma su una stringa a lunghezza fissa (impronta o message digest) ricavata dal messaggio stesso. Questo passaggio è effettuato per non gravare di costi computazionali l'esecuzione degli algoritmi di generazione e verifica, visto che un messaggio può essere di lunghezza variabile. L' algoritmo di hashing, impiegato nell'applicazione è l'SHA (Secure Hash Algorithm), sviluppato dal NIST (National Istitute of Standards and Technology). Si osservi come esso sia anche utilizzato dal governo degli Stati Uniti per produrre stringhe hash di 160 bit. In definitiva, sulla rete viaggeranno esclusivamente il messaggio in chiaro, la firma del mittente, e il certificato contenente la chiave pubblica, che saranno utilizzati per la verifica.

```
Prompt dei comandi
Microsoft Windows 2000 [Versione 5.00.2195]
(C) Copyright 1985-1999 Microsoft Corp.

D:\>keytool -export -alias marano -file marano.cer
Enter keystore password:  salvatore
Certificate stored in file <marano.cer>

D:\>_
```

Fig. 7: Estrazione del certificato in un file conforme allo standard X509.

Il processo di sottoscrizione, schematicamente mostrato nella Fig. 7, consta delle due operazioni sotto riportate:

- generazione dell'impronta del documento da firmare;
- generazione della firma mediante cifratura dell'impronta.

Al testo da firmare viene applicata una funzione di hash, appositamente studiata, che produce una stringa binaria di lunghezza costante pari a 160 bit. La funzione di hash assicura l'unicità di tale stringa, nel senso che a due testi diversi non corrisponde la medesima impronta, inoltre consente di evitare l'applicazione dell'algoritmo di cifratura all'intero testo, che può essre molto lungo e quindi inefficiente. La generazione della firma, come si evince dalla Fig. 7, consiste semplicemente nella cifratura, con la chiave privata, dell'impronta digitale generata in precedenza. In questo modo la firma risulta legata da un lato, attraverso la chiave privata usata per la generazione, al soggetto sottoscrittore, e dall'altro, per il tramite dell'impronta, al testo sottoscritto. Poichè il legame tra firma e documento, stabilito attraverso l'impronta, è di natura puramente logica, la firma stessa e le informazioni aggiuntive eventualmente ad essa associate possono essere registrate e gestite in modo del tutto separato rispetto al testo sottoscritto; in particolare possono trovarsi su supporti e sistemi di elaborazione del tutto indipendenti tra loro. Si è visto che per generare la firma digitale di un documento è necessario generare un hashing del messaggio da firmare e quindi codificare, l'hashing generato, con la chiave privata. Vediamo in dettaglio come sono state implementate queste due fasi. Per prima cosa si prepara la chiave privata, aprendo il file che la contiene e decodificandola.

```
FileInputStream keyfis = new FileInputStream(prk);
byte[] encKey = new byte[keyfis.available()];
keyfis.read(encKey);
keyfis.close();
PKCS8EncodedKeySpec privKeySpec = new
                              PKCS8EncodedKeySpec(encKey);
KeyFactory keyFactory = KeyFactory.getInstance(
                                            "DSA", "SUN");
PrivateKey privKey = keyFactory.generatePrivate(
                                            privKeySpec);
```

Per calcolare l'hashing del messaggio usiamo l'algoritmo SHA-1, mentre per la firma usiamo il DSA. Quindi si inizializza l'oggetto deputato alla firma impostando, attraverso i vari costruttori, tali algoritmi.

```
Signature dsa = Signature.getInstance("SHA1withDSA",
                                                  "SUN");
```

Successivamente bisognerà comunicare al dispositivo di firma la chiave con cui firmare il documento e il documento da firmare che ovviamente è contenuto in un file.

```
dsa.initSign(privKey);
FileInputStream fis = new FileInputStream(f);
 BufferedInputStream bufin = new
                                  BufferedInputStream(fis);
byte[] buffer = new byte[1024];
int len;
while (bufin.available() != 0) {
len = bufin.read(buffer);
dsa.update(buffer, 0, len);}
bufin.close();
```

Quindi si genera la firma e si salva quest'ultima in un file.

```
byte[] realSig = dsa.sign(); // firma del file
FileOutputStream sigfos = new FileOutputStream("Firma");
sigfos.write(realSig);
sigfos.close();
```

Assemblando tutte le parti precedentemente descritte, otteniamo l'oggetto JAVA che consente di generare la firma digitale di un documento informatico di qualsiasi tipologia.

LA VERIFICA

L'operazione di verifica della firma, mostrata nella

Algoritmi di hashing

Vengono utilizzati per testare la validità di un particolare messaggio. Gli algoritmi di hashing prendono in input una stringa a lunghezza variabile e la convertono in una stringa a lunghezza fissa.
I più diffusi sono:

- **SHA *(Secure Hash Algoritm)* sviluppato dal NIST a dal NSA;**
- **MD5 *(Message Digest Algoritm 5)* sviluppato da RSA.**

DSA

DSA *(Digital Signature Algorithm)* algoritmo che consente di generare la firma del documento.

firma digitale

Fig. 8, viene effettuata ricalcolando, con la medesima funzione di hash usata nella fase di sottoscrizione, il valore dell'impronta del messaggio trasmesso, e controllando che il valore così ottenuto coincida con quello generato per decodifica della firma digitale. In tal caso, il documento firmato dal mittente non ha subito modifiche ed inoltre si ha la certezza dell'identità del mittente il quale non potrà ripudiare il documento inviato.

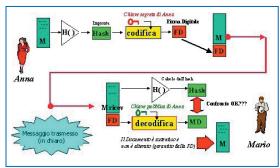
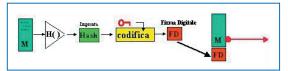



Fig. 8: **Processo di firma digitale.**

Per facilitare l'operazione di verifica, il mittente invia al destinatario, oltre al documento firmato e la relativa firma memorizzata in un file, anche il certificato relativo alla propria chiave pubblica. Inizialmente l'applicazione provvede ad estrarre il certificato di verifica dal file che lo contiene e successivamente provvede all'estrazione dal certificato della chiave pubblica, con la quale verificare il documento.

Fig. 9: **La firma del documento**

```
FileInputStream certfis = new FileInputStream( certPbk);
CertificateFactory cf = CertificateFactory.getInstance(
                                                 "X.509");
Certificate cert = cf.generateCertificate(certfis);
 certfis.close();
PublicKey pub = cert.getPublicKey();
```

Quindi si bufferizza la firma contenuta in un file, per la verifica della sua autenticità.

```
FileInputStream sigfis = new FileInputStream(firma);
sigToVerify = new byte[sigfis.available()];
sigfis.read(sigToVerify);
sigfis.close();
```

Per completare il set di dati necessari al processo di verifica, bisogna estrarre il documento da verificare dal file che lo contiene e memorizzarlo, in modo tale da renderlo compatibile con il formato che l'oggetto di firma, deputato anche alla verifica, si aspetta.

```
FileInputStream datafis = new FileInputStream(f);
BufferedInputStream bufin = new
                            BufferedInputStream(datafis);
byte[] buffer = new byte[1024];
int len;
 while (bufin.available() != 0) {
          len = bufin.read(buffer);
          sig.update(buffer, 0, len);
 };
bufin.close();
```

Disponendo delle informazioni necessarie, il processo di verifica può essere implementato. Bisognerà effettuare nuovamente l'hashing sul messaggio trasmesso e verificare che questo coincida con quello ottenuto dal processo di decodifica della firma. Visto che l'algoritmo di hashing usato in fase di firma è l' SHA-1 e quello usato per la firma è DSA, bisogna inizializzare l'oggetto che effettua la verifica con tali algoritmi e fornire ad esso anche la chiave pubblica con cui effettuare la decodifica della firma.

```
Signature sig = Signature.getInstance("SHA1withDSA",
                                                  "SUN");
sig.initVerify(pub);
Boolean verify = sig.verify(sigToVerify);
```

La verifica della firma avviene usando il *metodo verify* che, ricevendo come argomento la firma, restituisce un valore di verità che denota se la verifica ha avuto esito positivo oppure è fallita. Assemblando opportunamente le parti precedentemente descritte, otteniamo l'oggetto JAVA che consente di verificare contemporaneamente l'integrità di un documento ricevuto e l'identità del mittente garantendo, altresì, la non ripudiabilità di un documento inviato.

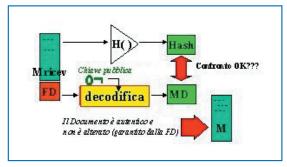



Fig. 10: **La verifica del documento firmato.**

Per avere una visione complessiva dell'applicazione, in articoli successivi, svilupperemo la GUI (Graphic User Interface). In modo tale da fornire al lettore la possibilità di realizzare un " provider di firma " per la certificazione mediante firma digitale di tutta la documentazione informatica di suo interesse.

Salvatore Marano

RSA

RSA (Rivest-Shamir-Adleman algorithm) algoritmo che consente di generare la firma del documento.

Crittogramma

Rappresenta il testo cifrato, ottenuto codificando il testo in chiaro con la chiave di codifica.

Crittoanalista

Figura professionale che si occupa della progettazione di tecniche e algoritmi da usare per la violazione di sistemi di crittografia.

Design

PATTERNS E APPLICAZIONI WEB

Design Patterns

Le Servlet e le Java Server Page sono gli elementi su cui viene a determinarsi il FRONT-END di un applicativo J2EE: per facilitare lo sviluppo di web application, portali o webservice si può ricorrere a precisi elementi di design.

Lo sviluppo di *software server-side* ha subito nel tempo un'evoluzione importante passando da script spesso non scalabili, come nel caso dei *C.G.I.*, a soluzioni eleganti come ASP ed il potente connubio JSP/Servlet. Le nuove potenzialità offerte non garantiscono da sole la creazione di software scalabile, manutenibile e performante: l'uso di design pattern aiuta lo sviluppatore a giungere più facilmente a questi obiettivi. I due design fondamentali, riscontrabili sia nelle JSP sia nelle Servlet, sono legati a dei meccanismi per il controllo del flusso applicativo: si parla di *Forwarding* e *Including* Pattern. Nel primo caso (*forward*) abbiamo un elemento (*chiamante*) che invoca il servizio di un altro elemento (*chiamato*), spostando a quest'ultimo il controllo del flusso esecutivo.

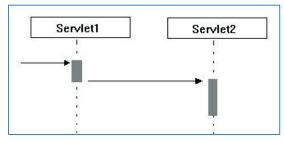



Fig. 1: **Diagramma di Sequenza Forward Pattern.**

Questa situazione è esprimibile, in una vsione statica, con il diagramma di Fig. 2.

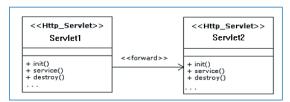



Fig. 2: **Diagramma di Classe Forward Pattern.**

Nel secondo caso (*include*) il controllo non viene perso dal chiamante: quest'ultimo dopo avere invocato e ricevuto i risultati dell'elaborazione del chiamato riprende la propria esecuzione.

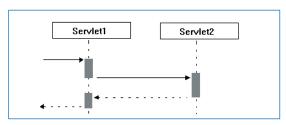



Fig. 3: **Diagrmma di sequenza Include Pattern.**

La diagrammazione statica sarà nella forma mostrata in Fig. 4.

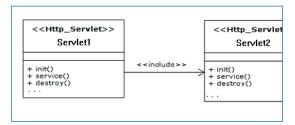



Fig. 4: **Diagramma di Classe Include Pattern.**

La possibilità di applicare queste logiche in combinazioni differenti di Servlet e Server Page fornisce flessibilità e possibilità notevoli per lo sviluppo di web application. E' bene, a questo punto, ricordare che entrambe le tecnologie presentate permettono di sfruttare a pieno le caratteristiche del linguaggio Java sul lato server: questa idea può trarre in inganno e portare alla nascita di applicativi poco scalabili ed ancor meno manutenibili, come nel caso di architetture *PAGE-CENTRIC* per le JSP. Nessuno vieta l'uso di Java Server Page secondo questa modalità, sebbene i risultati su progetti di grosse dimensioni possano rendersi scadenti: la massiccia commistione di porzioni esplicite di codice Java, tag html e tag, più o meno custom, delle JSP porta ad una disastrosa entropia nell'artefatto! Le Servlet possono proporre la stessa problematica: la necessità, ad esempio, di gestire per via programmativa il risultato visivo (rendering via HTML) può facilmente condurre ad un codice soggetto a staticità, impossibilità di riuso e difficile manutenzione. E' necessaria, allora,

Model View Controller

L'idea fondamentale è quella di dividere esplicitamente i ruoli di interfaccia (view), di logica di controllo dei dati e del flusso di lavoro (controller) e di modello dei dati o degli elementi di dominio (model).

Design Patterns

Fig. 5: Architettura PageCentric.

una suddivisione dei compiti che permetta di strutturare le risorse in modo da ottenere un insieme più efficace: per questo obiettivo è possibile sfruttare il pattern architetturale M.V.C. (*Model View Controller*). L'idea fondamentale è quella di dividere esplicitamente i ruoli di interfaccia (view), di logica di controllo dei dati e del flusso di lavoro (controller) e di modello dei dati o degli elementi di dominio (model). Adottare una tale differenziazione permette di raggiungere flessibilità, scalabilità e riuso nell'artifatto: per quest'ultimo punto si pensi, ad esempio, alla possibilità di adottare una politica di templating per la parte *view*. La logica presentata necessita di un ulteriore tecnologia per poter modellare i concetti fondamentali del nostro dominio: i JavaBean come elementi base per la realizzazione di Componenti non enterprise, permettono di ottenere quanto cercato. L'applicazione del paradigma *M.V.C.*, in pratica, porta ad una architettura di riferimento in cui le viste sono espresse da JSP, i controller sono Servlet e i modelli sono JavaBean.

MVC in pratica

L'applicazione del paradigma M.V.C. porta ad una architettura di riferimento in cui le viste sono espresse da JSP, i controller sono Servlet e i modelli sono JavaBean.

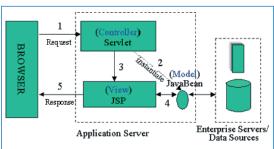



Fig. 6: Architettura (Model 2) di riferimento.

PRESENTATION PATTERN

La progettazione di applicativi web richiede uno studio attento delle caratteristiche che l'artefatto deve presentare per incontrare le esigenze del committente. Le considerazioni possibili e le problematiche da contemplare sono molteplici ed è facile perdere il controllo del processo di sviluppo. In quest'ottica sono d'ausilio una serie di elementi di design che con pochi concetti ci instradano sulla realizzazione di un artefatto robusto. Possiamo distinguere quattro pattern fondamentali: *decorating filter, front controller, dispatcher view* e *view helper*.

DECORATING FILTER: L'obiettivo di questo design è l'applicazione di uno o più filtri tanto all'oggetto di richiesta quanto a quello di risposta, rispettivamente all'ingresso o all'uscita dal web container. La presenza di filtri risponde all'esigenza di preprocessare i dati onde ottenere un'opportuna formattazione, sottoporli a validazione o semplicemente monitorare, con azione di *logging*, l'attività degli utenti. Il trend dei web services o di prodotti *xml-centric* rende tale preprocessamento importante, permettendo anche sensibili trasformazioni dei dati.

Nell'attuale specifica di riferimento per le Servlet e JSP, il concetto di *filtering* è presentato come elemento nativo delle web component J2EE secondo una logica di *chaining* dei filtri che fornisce una grossa flessibilità nella realizzazione di questo design.

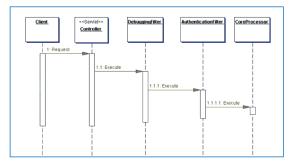



Fig. 7: FilterChain

FRONT CONTROLLER: Il design in questione prevede, nell'attuazione del paradigma *M.V.C.*, la presenza di un Controller per gestire il flusso esecutivo principale della web application; tale controller è l'iniziale punto di contatto per *l'handling* delle richieste verso il sistema.

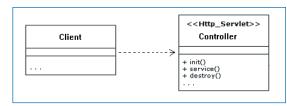



Fig. 8: Rapporto tra Client e Controller

Nel suo funzionamento, il front controller può delegare alcune elaborazioni ad alcuni *helper*, generalmente dei *JavaBean*, grazie ai quali è possibile, ad esempio,

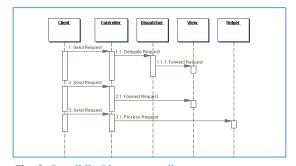



Fig. 9: Possibile Diagramma di sequenza per FrontController Pattern.

autenticare un utente. Altri elementi di ausilio possono essere utilizzati per gestire informazioni contenute nella *request* e facilitare l'operatività del *controller* che dopo aver svolto i propri *task*, passa il controllo del flusso esecutivo ad un *dispatcher*.

DISPATCHER VIEW: L'obiettivo di questo design è quello di portare il flusso di lavoro alla visualizzazione dell'opportuna JSP, in quanto espressione della view nell'architettura di riferimento. Questo pattern è un elemento composto che associa il concetto di *dispatcher* a quello di *view helper.* Un dispatcher è responsabile della gestione delle viste e della navigazione dell'eventuale sito o portale. Questo elemento può essere una porzione del front controller in grado di fornire un meccanismo di *dispatching* statico o può presentarsi come un componente separato atto a svolgere operazioni di smistamento dinamiche e più complesse. Una pratica comune è quella di affiancare questo pattern ad un classico elemento dell'insieme dei *G.o.F. design pattern* cioè il *COMMAND PATTERN.*

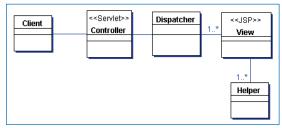



Fig. 10: DispatcherView Pattern.

VIEW HELPER: Questo pattern riguarda l'utilizzo di elementi di ausilio alle Java Server Page, onde ottenere un'adeguata formattazione dei dati in uscita o elaborazioni che possono spaziare dall'implementazione di politiche di *caching* o *templating* fino ad un'azione di *adapting* verso gli elementi di business (ad esempio si parla di *business delegate*).

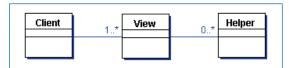



Fig. 11: ViewHelper Pattern

Da un punto di vista implementativo tale design si esprime nell'adozione di appositi JavaBean o nella creazioni di *Custom Tag*. Questi elementi possono essere usati separatamente anche se una maggiore pulizia realizzativa è ottenibile con il giusto connubio tra bean e tag: una prassi comune è, infatti, la creazione di JSP tag per la manipolazione di Componenti di formattazione e visualizzazione dotati di comportamenti complessi.

COMMAND: Grazie a questa soluzione di desing è possibile incapsulare in una famiglia di oggetti i possibili comandi d'interesse per il processo elaborativo di un'applicazione. I vantaggi nell'adozione di un tale ap-

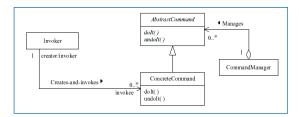



Fig. 12: Command Pattern

proccio nascono non solo dalla presenza di elementi parametrici ma anche dalla possibilità di applicare politiche di *undo* e *redo*, particolarmente sentite in prodotti di *text* o *graphical editing*. Una visione d'insieme, nell'uso dei principali *presentation pattern* è riassumibile nel seguente schema:

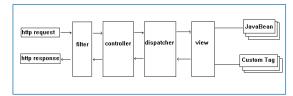



Fig. 13: Visione d'insiem dei Presentation Pattern.

UN PICCOLO FRAMEWORK M.V.C.

Esaminiamo delle porzioni di codice per concretizzare, anche se solo in parte, quanto esposto in termini concettuali utilizzando Servlet, JSP e JavaBean: costruiamo il *controller* che nella nostra visione delle cose sarà una specializzazione della classe *javax.servlet.http.HttpServlet*

```
public class Controller extends HttpServlet{
// dichiarazioni campi privati
 public void init(){
  // inizializzazione del controller
}
public void doPost(HttpServletRequest req,
 HttpServletResponse res) throws ServletException,IOException {
 RequestProcessor processor = new RequestProcessor(req);
Action action = processor.getAction();
   try{
        String nextView = action.execute(req,res);
       }catch(Exception exc){
            // politica di gestione degli errori sulle action
             ActionErrorManager.manage(exc);
         }
 dispatchToView(nextView, req, res); }
public void doGet(HttpServletRequest req,
                          HttpServletResponse res) throws
                               ServletException,IOException{
  // callback di processamento analoga alla doPost
}
private void dispatchToView(String view,
        HttpServletRequest req, HttpServletResponse res){
  RequestDispatcher dispatcher =
                         req.getRequestDiaspatcher(view);
```

Design Patterns

Sul Web

Una risorsa preziosa:
www.theserverside.com/patterns/index.jsp

Teoria e tecnica:
www.mindspring.com/~mgrand/pattern_synopses.htm

Per essere sempre aggiornati:
www.sdmagazine.com/uml/

In Italiano:
www.ugolandini.net/ExpertPattern.html

Design Patterns

ObjectWay

ObjectWay S.p.A è una società di consulenza tecnologica che opera dal 1990 realizzando soluzioni IT innovative per i settori delle Banche, Finanza, Telecomunicazioni, Industria e Pubblica Amministrazione. Vanta una vasta esperienza nella realizzazione di soluzioni basate sulle architetture J2EE, Microsoft .NET, Web Services e nella consulenza su Rational Unified Process.
I clienti sono aziende primarie tra cui Rasbank, Banca Popolare di Milano, Banca Sella, IntesaBCI, Banca Popolare di Bergamo, Ericsson, Alcatel, Gruppo FinMek, Tecnocasa, Commissione Europea. ObjectWay opera in Italia con uffici a Milano, Roma, Torino e Ispra (VA), con oltre 100 persone ed un volume di affari di circa 7 Milioni di Euro.
www.objectway.it

```
  if(dispatcher!=null) dispatcher.forward(req,res);
}
. . .
// seguono le dichiarazioni per eventuali altri metodi del
                                              controller
. . .
 public void destroy(){
  //operazioni di clean dell'ambiente!
}
}
```

È necessario evidenziare diversi elementi: il flusso esecutivo principale passa dalle fondamentali *callback* previste per le istanze della classe *HttpServlet* con la particolarità che l'elaborazione, come ultimo step, attua il *forwarding* verso la vista su cui verranno renderizzati i risultati del processamento. In questa fase ci sarà anche il coordinamento con altri metodi o elementi d'ausilio per attuare logging o l'autenticazione dell'utente. Nella callback di *init()* sarà contenuto il codice necessario, ad esempio, a ottenere dal container *datasource* o comunque altre fonti dati opportunamente configurate sul file *web.xml*. In maniera simmetrica la callback *destroy()* attuerà tutte quelle operazioni necessarie alla chiusura di risorse importanti, come nel caso di un collegamento ad un database remoto. Si noti che il concetto di *dispatcher* è espresso semplicemente da un solo metodo nel quale si sfrutta la classe *javax.servlet.RequestDispatcher* per attuare una politica di forwarding verso l'opportuna Java Server Page. L'altro elemento da evidenziare è la realizzazione del pattern Command ad opera di un elemento helper chiamato *Action*: otterremo un buon livello di flessibilità definendo il contratto minimale di una Action in un'interfaccia dedicata, mentre lasceremo a specifici JavaBean il compito di realizzare azioni concrete nel rispetto dell'interfaccia determinata. Un'ipotesi realizzativa può essere la seguente:

```
public inteface Action extends java.io.Serializable{
public String execute(HttpRequest req, HttpResponse
                              res) throws Exception;
public Object getModel(); }
public class SubmitAction implements Action{ …}
public class LoginAction implements Action{ … }
```

Ulteriore flessibilità è ottenibile isolando il processo di creazione delle Action in un'appositá Factory:

```
public abstract class ActionFactory{
  private static final ActionFactory instance =
                                    new ActionFactory();
  // costruttore privato per ottenere un singleton
  private ActionFactory(){}
  public static getInstance(){
    return instance; }
 public Action createAction(String actionType){
     // logica di creazione delle differenti Action  }
}
```

Il processo di creazione delle Action, nel rapporto con il controller, si completa nella classe *RequestProcessor*:

```
public class RequestProcessor{
private HttpRequest request;
  public RequesrProcessor(HttpRequest req){
    request = req;  }
  public Action getAction(){
   String actionType = String.valueOf(
                       request.getParameter("ACTION"));
   ActionFactory actFactory = ActionFactory.getInstance();
   return  actFactory.createAction(actionType); }
}
```

Con questi semplici elementi siamo nella condizione di poter definire, nella barra di navigazione del browser, una richiesta nella forma:

```
http://myhost:myport/servletPath/Controller?ACTION=
                               submit&attribuoX=10&...
```

La rappresentazione del modello, per completare la logica del paradigma M.V.C., sarà espressa da un insieme di JavaBean che realizzeranno i principali elementi del dominio del problema. In quest'ottica un dato *Component* Java può essere una semplice classe o un oggetto più complesso come nel caso del *pattern D.A.O.* (Direct Access Object): lo specifico JavaBean ha la capacità di sfruttare direttamente un *datasource* per manipolare i dati del back-end; generalmente si tratta di operazioni con un database remoto. L'ultimo desing esposto può interessare anche le Action che possono diventare veri e propri comandi per la gestione dei datasource. Un semplice esempio di *model-bean* con applicazione di desing *D.A.O.* può essere il seguente:

```
public class CarrelloDeiLibri implements java.io.Serializable{
private int numeroLibriPrenotati;
private ArrayList titoliLibri;
// altri campi privati...
  //getter e setter per gli attributi del bean
  public int getNumeroLibri(){ … }
  public void  setNumeroLibri(int num){… }
  public Collection getTitoli(){ … }
  public void setTitoli(Collection titoli){ … }
  public void datiDaRichiesta(HttpServletRequest req){
     // usa i setter sfruttando i dati presenti come
                                   attributi della richiesta}
 public void aggiorna(){
  // uso un datasource per aggiornare i dati sul DB }
 public void salva(){
  // rendo persistende l'attuale carrello dei libri }
 public void carica(){
  // uso i setter per ripristinare lo stato di un
                               particolare carrello dei libri
  // presente sul db.  }
}
```

Stefano Fago,
Java Consultant di ObjectWay SpA

Junit

Extreme Programming

ABBRACCIARE IL CAMBIAMENTO

Nelle prime due puntate di questa serie abbiamo scoperto come possiamo scrivere codice solido senza soffrire. Ancora non vi basta? Allora seguiteci in questa terza e ultima puntata, dove affronteremo insieme come eroici cavalieri, il più temibile drago del mondo della programmazione: il Cambiamento.

File sul CD
\soft\codice\ Extreme Programming

Sul CD allegato alla rivista troverete la libreria JUnit e tutto il codice sorgente di questo articolo.

Perché è così difficile programmare? Per dare una risposta completa a questa domanda non basterebbe tutto l'articolo. Ma se dovessimo selezionare un motivo tra tutti diremmo forse che programmare è difficile perché i programmatori inseguono un bersaglio mobile: le specifiche del software, che quasi sempre cambiano mentre ancora lo stiamo scrivendo. Le esigenze del cliente (e le nostre) cambiano velocemente, e ci tocca andare a correggere e modificare continuamente quello che abbiamo già fatto. Possiamo lamentarci, ma è così.
Quando le cose cambiano in continuazione, anche i piani più accurati hanno la triste tendenza ad andare a rotoli. Per questo motivo abbiamo sviluppato pratiche e tecnologie per scrivere software che sia anche più facile da modificare (la programmazione a oggetti è una di queste). Ma ancora non basta. Un sistema può essere pulito, elegante e ben progettato, ma quando lo modifichiamo ci esponiamo comunque al rischio di introdurre dei bug.
Cosa c'entra tutto questo con i nostri test unitari? C'entra, perché una delle caratteristiche dei test unitari è proprio quella di rendere facile e indolore il cambiamento. Anzi, potremmo dire che il motivo più importante per scrivere i test non è quello di verificare se il codice funziona ora, ma se funzionerà ancora in futuro dopo che lo avremo modificato. Finora non vi avevamo parlato di questo aspetto degli unit test perché sapevamo che molti di voi sarebbero stati scettici.
Ma ormai ne sappiamo tutti abbastanza per dimostrarvelo. Per questioni di spazio il nostro esempio sarà composto dalla solita semplice classe, quindi vi invitiamo a far funzionare la fantasia per immaginare queste tecniche applicate a sistemi molto più complessi. Allora, dove eravamo rimasti l'ultima volta?

SOFFIA VENTO DI CAMBIAMENTO

Nei due articoli precedenti abbiamo scritto una classe *Importo* che rappresenta dei movimenti di denaro su un conto corrente, e il relativo test unitario *TestImporto*:

```
/**
 *  Un importo in denaro.
 */
public class Importo {

   private final boolean _positivo;
   private final long _euro;
   private final long _cent;

   /**
    *  Costruttore per l'oggetto Importo
    *
    *@param  positivo  True se l'importo è in ingresso,
                                  false se è in uscita.
    *@param  euro  Importo in Euro (esclusi centesimi)
                                  - sempre positivo.
    *@param  cent      Centesimi - sempre positivo.
    */
   public Importo(boolean positivo, long euro, long cent) {
      _positivo = positivo;
      if(euro < 0 || cent < 0)
         throw new NumberFormatException("Valori
                negativi nel costruttore di un Importo");
      _euro = euro + (long)(cent / 100);
      _cent = cent % 100;
   }
```

```
/**
  * Restituisce l'importo come una stringa.
  *
  * @return   Il valore dell'importo.
  */
  public String getValore() {
    String risultato = "";
    if(!_positivo)
      risultato += "-";
    risultato = risultato + _euro + ".";
    if(_cent < 10)
      risultato += "0";
    risultato += _cent;

    return risultato;
  }
}

/**
 * Test case per la classe Importo.
 */
public class TestImporto extends TestCase {

  public static void main(String[] args) {
    junit.swingui.TestRunner.run(TestImporto.class);
  }

  /**
   * Test per il metodo getValore().
   */
  public void testGetValore() {
    Importo i1 = new Importo(true, 12, 06);
    assertEquals(i1.getValore(), "12.06");
    Importo i2 = new Importo(false, 0, 99);
    assertEquals(i2.getValore(), "-0.99");
    Importo i3 = new Importo(true, 12, 100);
    assertEquals(i3.getValore(), "13.00");
    Importo i4 = new Importo(false, 12, 375);
    assertEquals(i4.getValore(), "-15.75");
  }

  /**
   * Test per i valori negativi nel costruttore.
   */
  public void testCostruzioneSbagliata() {
   try {
    new Importo(true, -1, 0);
    fail("Mi aspettavo un'eccezione");
   } catch (NumberFormatException e) {}

   try {
    new Importo(true, 0, -1);
    fail("Mi aspettavo un'eccezione");
   } catch (NumberFormatException e) {}
  }
}
```

La classe è semplice, e il test conferma che è anche bella solida: basta far girare *TestImporto* nel *TestRunner* per ottenere una barra verde. Ora però abbiamo una nuova esigenza. Vogliamo aggiungere alla classe *Importo* un metodo che accetta in ingresso un secondo *Importo*, e restituisce in uscita un nuovo *Importo* che è la somma dei due:

```
public Importo somma(Importo i) {
 <...>
}
```

Naturalmente dovremmo anche scrivere un nuovo test per il metodo, ma la cosa non sembra particolarmente difficile. Anzi, tutto sembra molto semplice. Almeno fino a quando non ci proviamo. Il problema sta nel fatto che abbiamo scelto una rappresentazione interna piuttosto strana per la classe *Importo*. Il valore dell'importo è conservato in tre campi: uno contiene gli Euro (sempre positivi), uno i centesimi (anch'essi sempre positivi) e il terzo indica se l'importo è "in entrata" (positivo) o "in uscita" (negativo). Questa rappresentazione viene convertita in una stringa nel metodo *Importo.getValore()*. Ma per fare una somma non ci bastano due stringhe: ci servono due numeri.
Potremmo pensare di scrivere un metodo per la classe *Importo* che converte la rappresentazione interna in un numero, e poi usare quel metodo per fare la somma. Ma dovremmo poi chiamarlo per tutte le operazioni matematiche sugli importi, e la cosa potrebbe diventare noiosa. E a questo punto che senso avrebbe mantenere una rappresentazione interna così inutilmente complicata? Arrendiamoci all'evidenza: la scelta che abbiamo fatto nel primo articolo è stata un errore (va bene, ammettiamo anche che voi lettori non avete colpa). Dovremmo fare la Cosa Giusta. In questo caso la Cosa Giusta è cambiare la rappresentazione interna della classe.
Ma quando cambiamo l'interno di una classe rischiamo sempre di introdurre degli errori, quindi potremmo essere tentati di "mettere una pezza" al problema con il metodo di conversione in numero di cui parlavamo prima. Ma sbaglieremmo: in questo caso non dobbiamo preoccuparci più di tanto dei bug. Perché? Ma perché abbiamo i nostri test unitari, che diamine! Se cambiamo i campi privati e il resto dell'implementazione della classe Importo senza toccarne l'interfaccia, ci basterà far girare i test per verificare che tutto funzioni ancora. Non abbiamo scuse per non rendere il codice della classe più semplice e pulito.

LA SEMPLICITÀ PAGA

Anziché usare tre campi per contenere il valore di un *Importo* possiamo usarne uno solo. Ma attenzione: non facciamo l'errore banale di usare un tipo primitivo come un double per contenere il valore in Euro completo di decimali. Se usassi-

Junit

JUnit da scoprire

Questa serie di articoli riguarda i test unitari, non la libreria JUnit. Abbiamo scelto Java e JUnit perché sono semplici e molto diffusi, e anche perché JUnit è il primo e il più importante framework per gli unit test. Volevamo che questo corso servisse a tutti, non solo ai programmatori Java, quindi non siamo scesi in dettagli sulle caratteristiche avanzate di JUnit.
Ma se programmate in Java vi invitiamo caldamente a studiare la documentazione di questo piccolo grande framework. Tra le cose che dovete sapere: come usare i metodi *SetUp()* e *TearDown()* per impostare un test e "ripulire" dopo che il test è stato eseguito, e come costruire una "suite" composta da molti test, indispensabile per testare più di una classe. Se la cosa vi interessa, fatecelo sapere e scriveremo un articolo specifico su JUnit.

Junit

Test facili e test difficili

Scrivere test unitari è un'arte, o almeno artigianato. Gli esempi che abbiamo usato in questa serie sono semplici, e richiedono test semplici. Quando userete i test unitari lì fuori, nel mondo vero, le cose potrebbero non essere così ovvie, e sicuramente vi capiterà di trovarvi di fronte a problemi imprevisti. E' meglio mettere i test nello stesso package delle classi testate o da qualche altra parte? Come si fa a testare i metodi privati di una classe? Come si fa a testare un componente di rete, o un database? Come si fa a testare una classe che per funzionare richiede moltissimi parametri, che a loro volta sono configurati in qualche modo esoterico? Per alcuni di questi problemi esistono soluzioni semplici o comunemente accettate, per altri il territorio è tutto da esplorare. Anche di questo potremmo tornare a parlare in futuro, se qualcuno ci incoraggerà a farlo.

mo il valore double 4,20 per rappresentare una somma di denaro, la macchina eliminerebbe subito lo zero finale trasformando i nostri 4,20¤ in 4,2¤, scippandoci così 18 sontuosi centesimi. Una soluzione valida e semplice è invece quella di usare un unico long per conservare non gli Euro, bensì i centesimi totali:

```
private final long _cent;
```

In questo caso 4,20¤ sarebbero rappresentati dal numero 420. Abbiamo la stessa precisione che avevamo nella rappresentazione precedente (al centesimo). Non dobbiamo preoccuparci del segno, perché *_cent* può essere sia positivo che negativo. Ora dobbiamo modificare sia il costruttore che il metodo *getValore()* per adattarli a questa nuova rappresentazione interna. Ecco la nuova versione del costruttore, con gli stessi parametri di quello vecchio:

```
public Importo(boolean positivo, long euro, long cent) {
  if(euro < 0 || cent < 0)
    throw new NumberFormatException("Valori negativi
                              nel costruttore di un Importo");
  long centesimiSenzaSegno = euro * 100 + cent;
  if(positivo)
    _cent = centesimiSenzaSegno;
  else
    _cent = -centesimiSenzaSegno;
}
```

Abbiamo dovuto fare un po' di fatica per convertire i tre parametri in ingresso in un valore unico. Ora dobbiamo riscrivere l'implementazione del metodo *getValore()*:

```
public String getValore() {
  long euro = Math.abs((long)(_cent / 100));
  long centesimi = Math.abs(_cent % 100);

  String risultato = "";
  if(_cent < 0)
    risultato += "-";
  risultato = risultato + euro + ".";
  if(centesimi < 10)
    risultato += "0";
  risultato += centesimi;

  return risultato;
}
```

In questo caso il metodo è diventato un po' più complicato, perché abbiamo dovuto convertire i centesimi in Euro.
Fatto. Nel giro di pochi minuti abbiamo cambiato completamente l'implementazione della classe *Importo*. Tutto quello che resta del codice di poco fa è l'interfaccia. Una cosa come questa potrebbe darci qualche preoccupazione, ma per fortuna abbiamo i nostri fidati test. Dato che l'interfaccia della classe non è cambiata, i test non si sono nemmeno accorti di quello che è successo. Lanciateli, e dovreste vedere l'amata barra verde illuminare il vostro monitor e il vostro cuore. Successo!

GIOCARE CON I NUMERI

Ora che la rappresentazione interna dell'*Importo* è diventata semplicissima, possiamo scrivere facilmente il metodo *Importo.somma()*. Ma prima, in onore alla regola del "Test-First Design", scriviamo il test per questo futuro metodo. Per poter compilare il test ci serve però una versione "segnaposto" del metodo *Importo.somma()*.

```
public Importo somma(Importo i) {
  return null;
}
```

Ora possiamo scrivere il metodo *TestImporto.testSomma()*:

```
public void testSomma() {
  Importo i0 = new Importo(true, 0, 0);
  Importo i1 = new Importo(true, 10, 10);
  Importo i2 = new Importo(true, 12, 1);
  Importo i3 = new Importo(false, 2, 13);

  Importo somma1e2 = i1.somma(i2);
  Importo somma1e3 = i1.somma(i3);
  Importo somma2e1 = i2.somma(i1);
  Importo somma2e3 = i2.somma(i3);
  Importo somma0e3 = i0.somma(i3);

  assertEquals(somma1e2.getValore(), "22.11");
  assertEquals(somma1e3.getValore(), "7.97");
  assertEquals(somma2e1.getValore(),
                                    somma1e2.getValore());
  assertEquals(somma2e3.getValore(), "9.88");
  assertEquals(somma0e3.getValore(), "-2.13");
}
```

Forse è un po' lungo, ma ci sembra adeguato. Come al solito abbiamo cercato dei casi "difficili" nei quali il codice potrebbe non funzionare bene. Naturalmente il test per ora fallisce, perché non abbiamo ancora implementato il metodo *somma()*.
Ma con la nuova rappresentazione interna dell'*Importo* basta una riga di codice:

```
public Importo somma(Importo i) {
  return new Importo(this._cent + i._cent);
}
```

Facciamo girare il test e ammiriamo una bella barra verde.

ANDIAMO AVANTI?

Ora che abbiamo una rappresentazione interna migliore, ci rendiamo conto che il costruttore a tre argomenti della classe *Importo* è inutilmente complicato (sappiamo che molti di voi se ne erano resi conto già quando hanno letto il primo articolo di questa serie, ma all'epoca dovevamo ancora far finta di niente). Sarebbe meglio un costruttore con un solo argomento, che semplicemente prenda in ingresso il numero dei centesimi. Se volessimo eliminare il vecchio costruttore dovremmo cambiare l'interfaccia della classe, e questo significa che dovremmo modificare il codice delle classi che la usano. Dovremmo procedere in questo modo:

1) Scriviamo un nuovo costruttore con un solo argomento.

2) Testiamo il nuovo costruttore.

3) Modifichiamo tutte le chiamate al vecchio costruttore perché chiamino il nuovo.

4) Cancelliamo il vecchio costruttore (e il suo test).

In alternativa possiamo conservare il vecchio costruttore e semplicemente aggiungerne uno nuovo. Ecco un test per il nuovo costruttore ad un solo parametro:

```
public void testCostruttoreAUnParametro()
{
    Importo i1 = new Importo(1206);
    assertEquals(i1.getValore(), "12.06");
    Importo i2 = new Importo(-99);
    assertEquals(i2.getValore(), "-0.99");
    Importo i3 = new Importo(0);
    assertEquals(i3.getValore(), "0.00");
}
```

Ed ecco il nuovo costruttore:

```
private Importo(long cent)
{
  _cent = cent;
}
```

CAMBIARE FA BENE

Quando un programmatore deve modificare il codice che ha già scritto, tende di solito ad essere conservatore. Modifica meno codice possibile, per paura di rompere qualcosa. Questo significa però che le sue modifiche rovinano progressivamente il codice, rendendolo sempre più complicato.
A volte vorremo riscrivere intere sezioni del codice, perché sappiamo che sono ormai diventate dei generatori di bug. Ma non possiamo farlo perché c'è sempre poco tempo, e perché il rischio di introdurre errori è troppo grande. Così ci rassegniamo a convivere con i bug per paura di introdurne altri peggiori.

Figura 1 **L'interfaccia di JUnit in azione**

Nell'esempio di questo mese abbiamo capito che il nostro codice era troppo complicato, e abbiamo cercato di renderlo più semplice. Abbiamo cercato, come dicono i fan dell'Extreme Programming, di fare "la cosa più semplice che potesse funzionare". I test ci hanno permesso di modificare il codice velocemente e in piena sicurezza. Se un sistema è completamente coperto da test unitari, allora non importa quanto è complicato: potremo sempre modificarlo alla massima velocità possibile. Una barra verde ci garantirà che tutto funziona ancora, oppure una barra rossa attirerà la nostra attenzione su quello che non funziona più. E' incredibile la quantità di modifiche che si possono fare in poco tempo quando ci sono i test. A noi è capitato di cambiare completamente le parti più interne e delicate di programmi molto complicati nel giro di poche ore.
Provateci e crederete.

CONCLUSIONI

E con questo siamo arrivati alla fine della nostra breve serie su JUnit. Speriamo di avervi messo una piccola pulce nell'orecchio. L'utilità dei test unitari non è affatto intuitiva, quindi molti programmatori rifiutano questa tecnica pensando che si tratti di una bizzarria. Ma di solito chi la prova per un po' di tempo finisce per non abbandonarla più. E così i test unitari stanno diventando sempre più importanti, e quella che due anni fa era una stranezza per eccentrici pionieri sta ormai diventando una delle tecniche più celebri e universalmente valide dell'ingegneria del software. Forse in futuro torneremo ancora su questi argomenti. Tenete d'occhio queste pagine. In ogni caso ora avete un po' di strumenti nuovi nella vostra cassetta degli attrezzi. A voi il compito di usarli al meglio.

Paolo Perrotta

Questo è solo l'inizio

Se siete pronti a saperne di più e conoscete abbastanza bene l'inglese, andatevi a leggere i tanti articoli sugli unit test pubblicati sul sito ufficiale di JUnit

http://www.junit.org

L'esperto risponde...

Puntatori fantasma

Egregia Redazione, sono un lettore affezionatissimo di ioProgrammo fin dai primissimi numeri e non ho mai smesso di apprezzare il vostro encomiabile lavoro. Vi scrivo per sottoporvi un quesito: sto scrivendo un'applicazione in Visual Basic e mi farebbe comodo far scomparire il puntatore del mouse in alcuni momenti. Ho trovato diversi esempi su come cambiarne la forma ma non sono riuscito a capire se e come sia possibile farlo scomparire.
Sicuro di una vostra cortese risposta, vi auguro il meglio.

Riccardo Malagodi

Gentile Sig. Malagodi, in Visual Basic non ci sono comandi che direttamente realizzino quanto lei chiede: bisogna dunque "sporcarsi le mani" con le API. Supponiamo che nel suo Form ci sia un pulsante chiamato *Puntatore*, ecco il codice che consente alla pressione del pulsante di far sparire e poi riapparire il cursore:

```
Private Declare Function ShowCursor Lib
    "user32" (ByVal bShow As Long) As Long

Dim blnShowCursor As Boolean

Private Sub Puntatore_Click()
  ' Call the API and pass it the form-wide
                                    boolean.
  Call ShowCursor(blnShowCursor)

  ' Switch the form-wide boolean.
  If blnShowCursor Then
      blnShowCursor = False
  Else
      blnShowCursor = True
  End If

End Sub
```

Un piccolo consiglio: le conviene prevedere uno short-cut per il pulsante altrimenti, una volta sparito il puntatore le risulterebbe un po' difficile premere di nuovo il pulsante per farlo riapparire… glielo dico per esperienza!

Thread: i vantaggi

Gentile redazione, complimenti per la vostra interessante rivista! Da un pò di tempo sto realizzando delle applicazioni in java, ho letto qualcosa sui thread e vorrei utilizzarli ma prima vorrei capire bene quali sono i vantaggi derivanti dal loro utilizzo e come si creano. Grazie!

Luca

Uno dei vantaggi principali è la "leggerezza", infatti la loro creazione, distruzione e sincronizzazione, sono operazioni molto "economiche" grazie alla condivisione dello spazio di indirizzamento. Tuttavia, bisogna stare molto attenti ai problemi che possono scaturire dal loro utilizzo (mutua esclusione, sincronizzazione…)! Per quanto riguarda la loro creazione, il modo più semplice è quello di creare una sottoclasse di Thread e ridefinire il metodo *run*:

```
Class MiaClasse extends Thread {
  ….
  public void run () {
    …
  }
}
public class TestThread {
  public static void main (String args[]) {
    Thread t1 = new MiaClasse();
      t1.start();
    }
}
```

Si noti che *Class Thread.start* non fa altro che lanciare il metodo *run()* del thread.

Le Date in Java

Ciao a tutti! Sono un ragazzo di 23 anni ed ho intrapreso da poco la strada del programmatore java. Potreste essere così gentili da darmi dei chiarimenti circa l'utilizzo delle date?.Vi ringrazio in anticipo perché sono certo che mi risponderete al più presto!
Cordiali saluti

Fabio

Per la gestione della date, Java mette a disposizione gli oggetti *Date*, *Calendar*, e *GregorianCalendar*, naturalmente la scelta di uno o dell'altro dipende dalle specifiche esigenze. *Java .util.date* memorizza il tempo (espresso in millisecondi) trascorso dal momento in cui la classe viene istanziata. Potrebbe risultare utile, ad esempio, qualora fosse necessario calcolare il tempo d'elaborazione di un processo.
Java .util.Calendar consente di conoscere anche giorno, mese, anno, ora, ecc. in funzione di un certo tipo di calendario e fuso orario. Può essere utilizzata nel caso di gestione di archivi anagrafici, date di scadenza, ecc.
Java.util.GregorianCalendar ha le stesse finalità della classe *Java.util.Calendar* ma è specializzata per l'utilizzo del calendario gregoriano, usato nella maggior parte dei Paesi occidentali.

SQL e l'eliminazione dei duplicati

Cara redazione, da qualche giorno mi sto cimentando con la programmazione in SQL.
Ho notato che rispondete sempre in modo esaustivo alle domande

dei lettori, così ho pesato di porvi un mio quesito. Ho creato un database che contiene dei dati anagrafici, a me interessa sapere solo quali sono i nomi di queste persone, senza i duplicati. Quindi devo trovare il modo di “dire” ad SQL di non considerare le righe duplicate. Come fare? Grazie!

Giulia

Cara Giulia, il comando da usare è *“Distinct”*. In particolare, supponiamo che tu abbia creato la tabella *“Persone”* con le seguenti tre colonne: *codice, cognome, nome.*
La query che devi fare è la seguente:

```
SELECT DISTINCT nome
FROM Persone
```

Così ti verranno visualizzati tutti i nomi, diversi fra di loro, che compaiono nelle tabella *“Persone”*.

JDBC-ODBC

Salve! Sono una vostra affezionata lettrice, da un po’ di tempo mi sto dilettando a realizzare delle applicazioni in java. A tal proposito, ho un quesito da porvi:cos’è JDBC? E a cosa serve?
Vi ringrazio in anticipo!

Anna

Ciao Anna, JDBC sta per *Java Database Connettivity*, è rappresentato dal package *java.sql* e consente l’interfacciamento coi database. Per poter utilizzare questo package, è sufficiente un driver JDBC/ODBC. In effetti, JDBC non è altro che un adattamento di ODBC al linguaggio java.
Sfruttando i vari metodi che java mette a disposizione, possiamo far interagire le nostre applicazioni java con un qualsiasi DBMS. Vediamo un esempio:

Prima di tutto carichiamo il driver *JDBC/ODBC*:

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

Poi apriamo la connessione col database da noi precedentemente creato in Access:

```
dbconn = DriverManager.getConnection(
                    "jdbc:odbc:miodatabase");
```

Eseguiamo un’istruzione SQL:

```
Statement statement =
                    dbconn.createStatement();
ResultSet rs = statement.executeQuery(
                    "SELECT dati FROM tabel1a")
```

L’istruzione *Select* va a prelevare dei dati dal nostro database, i risultati vengono memorizzati nell’oggetto *rs*.
A questo punto, possiamo fare delle operazioni sui dati ottenuti (ad esempio possiamo stamparli o modificarli).
Per chiudere la connessione:

```
dbconn.close();
```

Biometria

Cara redazione di ioProgrammo sono un programmatore "IN ERBA" di 18 anni ed avrei bisogno del vostro aiuto: Nel numero 66 di febbraio ho trovato molto interessante l'applicazione in Visual basic di Biometria a "portata di mano" ma ho un grande problema appena mando in run l'applicazione il debugger mi da l'errore sul'evento *form_load* perchè non trova il File: *"USERS.DAT".* Come posso fare?

via mail

Risponde Elia Florio

Se guarda bene nell'avvio della form, trova il seguente codice:

```
Private Sub Form_Load()
    'setta il livello di sicurezza intermedio
    'per il riconoscimento impronte
    m_nSecuLevel = 2

    'legge il numero di utenti schedati in archivio
    'dal file C:\USERS.DAT
    Open "C:\users.dat" For Input As #1
    Line Input #1, n
    ID.Text = n
    Close #1
End Sub
```

Come può vedere, l'applicazione all'avvio apre in lettura il file *"C:\USERS .DAT"*, che dovrebbe contenere una riga col numero di utenti presenti nel sistema di riconoscimento.
Può benissimo creare questo file, usando NOTEPAD per creare un nuovo file di testo al cui interno è inserito il valore "0" e rinominando il file da *"USERS .TXT"* a *"USERS.DAT"*.

Java 3D

Ho letto con molto interesse gli articoli sulla modellazione 3D in Java.
Successivamente ho provato ad eseguire il codice. Tutto funziona correttamente ma i solidi vengono visualizzati deformati; infatti tutti i solidi hanno come vertice l'angolo superiore sinistro della finestra (ad esempio il cubo è centrato nella finestra ma ha il vertice superiore sinistro che corrisponde all'angolo superiore sinistro della finestra, quindi e' tutto allungato in quella direzione). Qual è il problema?
Grazie.
Cordiali saluti e complimenti.

Cristiano Zoffoli

Risponde Giuliano Uboldi

Caro Cristiano, per scrupolo sono andato a ricontrollare il codice per verificare che non contenesse errori e, compilando e lanciando l'esempio tutto è proceduto regolarmente. Sia il quadrato superiore che la piramide in rotazione che la linea retta sono visualizzati correttamente. Il problema che tu dici di avere, evidentemente dipende dal cattivo posizionamento di alcuni dei vertici dei solidi che hai generato. Mi sembra di capire che non hai eseguito il codice presente sul CD di ioprogrammo ma hai riscritto a mano il programma di esempio.
Se così fosse controlla accuratamente la posizione di ogni vertice o prova ad eseguire il codice fornito con il CD per verificarne il buon funzionamento.

Per contattarci:

e-mail: iopinbox@edmaster.it

Posta: Edizioni Master,
Via Cesare Correnti, 1 - 20123 Milano

Micro Java Network

http://www.microjava.com/

Attualmente, l'intera piattaforma Java 2 si divide in tre differenti edizioni, ognuna destinata ad un particolare settore. Al centro di tutto c'è la famigerata Standard Edition (J2SE), destinata ai desktop dei PC di casa, all'ufficio e ai portatili di recente fattura. Questa edizione permette lo sviluppo e l'esecuzione di applicazioni client destinate all'utente finale. Non a caso, in questa distribuzione si trovano i package AWT e Swing, che contengono ogni strumento utile per la costruzione di elaborate interfacce a finestre. Con la Standard Edition è possibile realizzare applicazioni stand-alone, applet e software destinati alla tecnologia Java Web Start. Le più recenti versioni dell'edizione, inoltre, includono il Java Plug-in, un componente che permette l'esecuzione delle applet all'interno dei più comuni Web browser. Alle imprese è invece dedicata la Enterprise Edition (J2EE). In questa confezione si trovano numerose classi concepite per arricchire e facilitare lo sviluppo di complesse applicazioni server. Giusto per citare alcune tra le più celebri tecnologie del lotto, qui trovano posto le API per XML, per SOAP e per CORBA. Ci sono anche i pacchetti di JavaMail (per lo scambio di e-mail), Java Servlets e JavaServer Pages (per documenti Web dinamici e per Web service), JDBC (per l'accesso alle basi di dati) ed altri ancora. La Enterprise Edition è tra le edizioni maggiormente apprezzate (Per .NET sarà dura insidiare il primato di Java nel settore!). Ultima arrivata, ma non per questo meno importante o meno diffusa, è la Micro Edition (J2ME). Grazie a questa confezione, Java varca la più recente frontiera dello sviluppo di applicazioni, piombando nel variegato universo del wireless, dei dispositivi portatili e degli apparecchi con ridotte capacità di calcolo. A sua volta, l'edizione si suddivide in diversi raggruppamenti di tecnologie, che coprono lo sviluppo per i palmari, per i telefoni cellulari, per le televisioni digitali, per le smart card e per altri dispositivi di questo genere. Questa edizione "portatile" di Java sta riscuotendo grande successo, soprattutto nel settore della telefonia mobile, che ultimamente vive un'espansione senza precedenti. Sono molti i produttori di telefoni cellulari che hanno deciso di dotare i propri prodotti di una Java Virtual Machine compatibile con le specifiche di J2ME. Tra le aziende più devote, spiccano Nokia e Sony-Ericsson. Tutto ciò significa che chi possiede un telefono Java-enabled può tranquillamente scaricare ed eseguire applicazioni concepite per la J2ME. A farla da padrone, in questo momento, sono soprattutto le applicazioni ludiche, i cosiddetti "giochini per il cellulare". La portabilità, ovviamente, è tra i maggiori fautori di questo successo. Il motto «scrivi una volta, esegui ovunque», con l'arrivo di J2ME, ha accresciuto la propria enfasi. Immaginate quanto sia comodo, per un programmatore, scrivere un videogioco che, senza bisogno di modifiche, possa essere eseguito su una vasta schiera di telefoni cellulari, di produttori diversi e con caratteristiche differenti. Prima dell'avvento di J2ME, questo risultato era utopia. Ogni cellulare disponeva del suo personalissimo kit di sviluppo. Trasportare un gioco o un'applicazione da un apparecchio ad un altro, richiedeva un ingente dispendio di risorse, passando spesso attraverso la completa riscrittura del codice. Adesso le cose sono più semplici. Tuttavia, per raggiungere risultati ottimali, non basta conoscere Java e documentarsi sulle API di J2ME: bisogna anche assorbire le particolari tecniche di sviluppo che consentono una programmazione efficiente e realmente multipiattaforma. Per il conseguimento di questo scopo, è bene procurarsi libri appositi e consultare incessantemente delle risorse Web specializzate, come Micro Java Network (*http://www.microjava.com/*). Il sito si propone nella veste di portale onnisciente verso le tecnologie comprese nella Micro Edition di Java. Dalla sezione "Downloads" è possibile prelevare applicazioni e giochi provenienti da tutti gli angoli del globo. Se vorrete, potrete inviare al sito le vostre creazioni, cosicché siano rese note alla comunità. Ad ogni modo, nel settore "*Downloads*" non si trovano soltanto i programmi completi: nell'angolo trovano posto anche le classi di utilità, le virtual machine, gli emulatori, gli strumenti di sviluppo e così via. La sezione "*Articles*" è tra le più appetibili, poiché dispensa utili informazioni per una buona programmazione. In "*Devices*" è mantenuto un archivio dei dispositivi compatibili con le specifiche della Micro Edition, con informazioni dettagliate su ogni differente soluzione. Nella sezione "*Publishing*" si spiega come fare in modo che i propri lavori raggiungano un gran numero di utenti finali, con informazioni sui canali di distribuzione e vendita. Alla voce "*Developer*" si trovano FAQ, link, presentazioni di libri ed altro ancora. Infine, non può mancare un raggruppamento di punti di discussione, dove i frequentatori del sito possono incontrarsi e scambiarsi consigli utili. Il sito è ben fatto, facile da navigare, bello da vedere, leggero. Per accedere alla maggior parte delle risorse, ad ogni modo, è necessario registrarsi. Ne vale la pena.

Carlo Pelliccia